

SyncWare News

The journal for technical
applications of T/S
computers

Volume 4 Number 1

Sept.-Oct. '86



Another Year Older

This issue marks the beginning of our fourth year of publishing SYNCWARE NEWS. While the past year included some changes within our publishing group, a new editor for this journal and starting up a separate publication dedicated solely to the QL, we feel proud of our efforts and accomplishments. We are planning to make Volume #4 even better.

Coming issues this year will include more material for the less technically inclined on the basics of T/S computing. We'll continue regular installments of Basil Wentworth's fine MC series for the novice programmer. The column "Basic Topics" that first appeared in 3:6 will continue as material is available. Articles are in the works for dedicated applications using your 1000, 1500, and 2068. So even if your faithful old Timex computer isn't your front line machine any longer, don't let it just set on the shelf and gather dust. Put it to work!

You "old" (ok, experienced?) software and hardware pros needn't feel left out. Folks like John Ollger, our own Fred Nachbaur and Tom Bent, and others are still out there, putting their tired little computers through the paces in an effort to bring top notch, hi-tech material about your machines to you. We also are planning another PROGRAMMERS COMPETITION!!! Complete details of the contest will be announced in a future issue. So "old timers", there's plenty of material to keep you folks busy too.

2nd Annual Fest Planned

The 1986 Midwest T/S Computer Fest chairman Frank Davis has announced that there will be a 2nd Annual Midwest T/S Computer Fest! Tentative dates for the grand event are May 2-3, 1987. The location has been changed to Indianapolis, IN. Plans include seminars, a dealers room, and a special banquet for dealers and users groups Friday night before the Fest. Information can be obtained from Frank Davis, 513 E. Main St., Peru, IN 46970. Please include a SASE.

TS Communications

One area of T/S computer use that needs to be further delved into is telecommunications. There is a wide variety of modems, interfaces, and terminal software currently available for all the T/S machines. The Indiana Sinclair Timex Users group is currently running a bulletin board, with software they've developed, on an unexpanded TS2068. The BB can be reached at (317) 898-3903. Set your moden for 7 bytes, 1 stop, and even parity. At present this is only a message type bulletin board. A future issue of SWN will have the group's complete story including how-to instructions for creating your own service. For now, get a modem, get connected, and try ISTUG's number.

FOR YOUR SUPPORT

This column announces any software, hardware, and related modifications that are new or otherwise untested by us. If you have something of interest, we will announce it here. Send us a description of your product (PLEASE keep it short). We advise readers to send a self addressed and stamped envelope, as a courtesy, to the individuals or companies for information.

The John Oliger Co., 11601 Whidbey Dr., Cumberland, IN 46229, is currently offering the Oliger 2068 Disc System. Available as bare boards, kits, or completed and tested units, prices range from \$17.95 to \$119.95 pp. The disk system consists of two boards which plug into the Oliger 2068 Expansion Board (not included). Features include ease of use, high speed, and Spectrum and TS2068 compatibility. Write for complete details and price info.

NovelSoft, 106 Seventh Street, Toronto, Ontario, Canada M8V 3B4, (416) 259-8682, announces two new pieces of software for either the TS2068 or Spectrum. ARTWORX allows you to not only draw, but paint and do graphic design on your computer. Features include pull-down menus, multiple brushes and text fonts, cut and paste, and much more. Includes sample artwork and manual. TIMACHINE is a BASIC compiler that will convert your programs into machine code to increase running speed to 200 times normal. The program will handle all BASIC and floating point except I/O. Includes demo programs and manual. Both programs include Spectrum and 2068 versions. The cost is \$19.95 each U.S. plus \$3 S&H.

Willcocks Research Consultants, 6321 W. 78th Place, Los Angeles, CA 90045, (213) 215-0780, announces the QL version of KARTIK, a family crossword game, which includes an electronic dictionary. KARTIK and TANGLE-4 are also available for TS1000/1500 computers. Write for details and prices.

Grey & Clifford Computer Products, P. O. Box 2186, Inglewood, CA 90305, (213) 759-7406, is offering software and hardware for communications on the Spectrum emulated TS2068. SPECTERM-64 is terminal emulation software which allows 64 column display without additional hardware. Price for SPECTERM-64 is \$30. When used with the new Z-SI/O hardware card, it allows use with any RS-232C modem or peripheral including Hayes compatibles. Price for the Z-SI/O is \$79.95. Write for more details.

Knighted Computers, 707 Highland Street, Fulton, New York 13069, is selling QL computers, software, and related products. Their current flyer lists the book "T/S2068 Basics and Beyond" by Sharon Z. Aker and the line of Intraclean Audio and Video Recorder Care Systems among some of the products they are currently handling in addition to computer hardware and software.

G. Russell Electronics, RD 1 Box 539, Centre Hall, PA 16828, (814) 364-1325, has their current catalog available upon request. Many hardware and software items are listed for the QL, 2068, and 1000 computers.

Jim Houston Interpises, 414 West Elsmere Pl., San Antonio, TX 78212, offers a shifter board for Timex computers. This allows you to use a full size keyboard and assign the extra keys with functions that normally require multiple key presses. The Shifter Board is sold bare for \$17.50 or fully assembled and tested for \$45.00. Also available are an Edge Connector board, \$55.00; a Direct Video with Inverter board, \$15.00; and a line of programs on AERCO disk, for \$29.95 each. For more information, please send a SASE to the above address.

Larry Kenny, Larken Electronics, has announced that he is working on a cart-bases DOS for the Larken 2068 disk drive. Features include Spectrum compatibility, built-in NMI "snapshot" save, usage of standard tokens, and disk compatibility with other systems. Also in the works are EPROMs for use with other popular disk systems. Contact LARKEN ELECTRONICS, RR#2, Navan, Ontario K4B 1H9, Canada, to be placed on the "inform-when-ready" list, or to suggest features that YOU would like to see.

Chia-Chi Chao, 73 Sullivan Drive, Moraga, CA 94556, announces that he is now selling three programs from JRC Software. Diamond Mike II, \$17.00; Great Game & Graphics Show, \$17.00; Compass, \$19.00. All prices include shipping. The games are available on tape or 5.25" Aerco disk. A complete catalog is available with SASE.

Fred Nachbaur, Silicon Mountain Computers, C-12, Mtn. Stn. Group Box, Nelson, BC V1L 5P1, Canada, has been hard at work under the mountain and has finally completed DUNGEON OF YMIR for the TS1500. This full featured HI-RES adventure game competes favorably with similar games for much larger machines. Presently available are V1 (requires Hunter board, other 8-16K NVM or 64K) and V2 (requires 16K RAM pack; may require slight hardware mod, supplied). V3 (for ZX81/TS1000 with CMOS 8-16K RAM) to follow soon; stay tuned. Price \$24.95 U.S.

Robert C. Fischer, producer of PRO/FILE EXTENSIONS, T/S GRADER, WORD PUZZLER, and WORD GAMES, has changed his address and can now be found at Rt. 2 Arizona St., Emerson, GA 30137

Bill Heberlein, 5052 N. 91st St., Milwaukee, WI 53225, is selling video tapes of the proceedings of the 1st Annual Midwest T/S Computer Fest. The cost is \$2.50 plus a blank tape in your format.

Bill Bell, 596 Cherrington Road, Westerville, OH 43081, (614) 882-3883, is offering BBDOS for the TS1000 equipped with the Aerco FD-ZX Disk Interface. BBDOS is a fully automatic, BASIC transparent, disk operating system that locates itself in the 8-16K region of RAM. It is advertised as being fully compatible with all printer interfaces. \$29.95.

C. W. Associates, 419 N. Johnson Street, Ada, OH 45810, have announced that they are now Authorised QL Dealers. Their new catalog is currently available for the asking.

FORUM

SYNCWARE NEWS welcomes correspondence from its readers. Some editing of letters may be necessary for brevity or to enhance clarity.

Dear Editor,

I'd like to thank SWN and Mr. E. Swallow for sharing his solved problem. His recommendation about keeping your cassette recorder away from the left side of the computer when doing a save solved a problem I've been having for almost a year.

Marlin Perkins
Denton, Texas

Screen Hash

Dear Editor,

As owner of the Zebra Disk System for the 2068, I am perplexed as to why my monitor (a Comrex 5650) intensity and sharpness deteriorates whenever a parallel printer interface is attached to the Zebra Expansion interface. This occurs with the Tasman, Aerco, and the JLO printer interfaces. In addition, Tasword II is almost unreadable.

If the printer interface is removed, the problem disappears. Similarly, if the printer I/F remains and the disk drive interface is removed, the problem also disappears. Having the 2040 printer connected to the bus apparently has no effect either way.

What could be the culprit pulling down the video and how can I eliminate it?

Barry Carter
Warren, MI

Dear Mr. Carter,

Since none of us at SWN have the Zebra Disk System, I'm afraid I can't address your problem from direct experience. We also haven't heard from anyone else with similar problems. However, I think I might be able to help isolate your problem.

The 2068 video is derived from a LM1889N color modulator chip. This is completely isolated from the CPU data, address, and control lines which are accessed by the Zebra interface. Unlike the ZX81/TS1000, where the video is obtained from a master chip, the 2068's video signal is generated by a separate device that has nothing to do with the computer.

There is really only one thing this chip has in common with the rest of the system: THE POWER SUPPLY. Power to the video chip is obtained through a little 12 volt regulator about the size of a small transistor. This is U8, a 78L12. If your supply voltage drops too low because of the addition of peripherals, this could cause your difficulty. Adding the interface might just be the "straw that broke the camel's back". I find this especially likely in light of the fact that three different interfaces all show the same phenomenon.

Other things to check for: Try moving your monitor cable relative to the printer cable, power supply wire, and cassette cables and see if this makes any difference. Contact Zebra and describe your problem. Possibly others have reported similar difficulties. Perhaps you have a unit which for some reason draws

too much current. If your monitor has a HI-LO impedance switch, see if its setting makes any difference (unlikely, but check it anyway; if there's a difference, it could point to a bad driver transistor, Q2 and/or Q3). Are your monitor, printer, and disk drive chassis properly grounded? Are you close to any powerful TV or radio stations? (If so, a power-line "RFI filter" might help.) Try a different video cable.

Please let us know if any of this helps. If you find the problem fill us in so we can let others know.-fn

Printer Squash

Dear Editor,

My 2040 printer seems to print in squashed letters until about 2 feet of paper has printed out. This is especially true when I leave the printer on even when I turn my computer off. Is there an easy fix for this?

Also, There are some small hand held computers to translate English to Italian and German. Are there any programs designed for the 2068? I've tried routines like:

```
10 LET YES=SI
20 PRINT "YES"
```

I just get a "variable not found" error. What do I need to do?

Sergio Frost
San Diego, CA

WOW! AT LAST!! A VERY AFFORDABLE COMPUTER AT A VERY AFFORDABLE PRICE

POWERFUL FULLY PROGRAMMABLE WITH 2K OF MEMORY—PORTABLE—6.7" x 1.38" INCH MODULE SINGLE KEY ENTRY COMMANDS—DURABLE 40 KEY MEMBRANE TYPE KEYBOARD—280A BASED FOUR CHIP DESIGN—EDUCATIONAL—UNIQUE SYNTAX CHECK REPORT CODES FOR ERROR IDENTITY—GRAPH DRAWING AND ANIMATED DISPLAY—ACCURATE TO 9-1/2 DECIMAL PLACES FOR FULL RANGE MATH AND SCIENTIFIC FUNCTIONS—AT AN AFFORDABLE PRICE

WE CANNOT TELL YOU THE MAKE OF THE COMPUTER BUT IT WAS MADE BY A FAMOUS WATCH COMPANY THEY USED TO SELL FOR \$99.95

WE BOUGHT OUT WHAT THE FACTORY HAD LEFT IN STOCK AND HAD TO REMOVE THE LABELS, THESE UNITS ARE UNPACKAGED LESS THE 9V WALL ADAPTER AND MANUAL BECAUSE THIS IS A DISCONTINUED ITEM THERE IS NO WARRANTY

LIMITED SUPPLY

GET THEM WHILE THEY LAST

BUY 1st UNIT FOR \$19.95	BUY 2nd FOR \$16.95	9V DC WALL ADAPTOR	\$4.95
BUY THE 3rd UNIT (NON OPERATING FOR PARTS) \$10.95		MANUAL (OVER 150 PAGES)	\$2.95

See September 1984 issue of 73 for TIMEX/RTTY article

CHIP BONANZA (AT THESE PRICES THEY ARE A STEAL)

2708		\$1.00 EA OR 10 FOR \$ 9.00
2718		\$2.25 EA OR 10 FOR \$20.00
2732		\$3.25 EA OR 10 FOR \$30.00
2784		\$4.00 EA OR 10 FOR \$35.00
27138		\$7.00 EA OR 10 FOR \$60.00
8902		\$4.95 EA OR 10 FOR \$45.00
8810	(REG \$3.95)	\$1.95 EA OR 10 FOR \$18.00
88A09	(REG \$19.95)	\$5.95 EA OR 10 FOR \$50.00
88A21	(REG \$9.95)	\$2.95 EA OR 10 FOR \$25.00
4118	(or equivalent)	8 FOR \$ 5.90
4134		9 FOR \$12.95
XR 2211	(SPECIAL)	\$ 2.95
TMS 9901NL MICRO-P 84 PIN 8 BIT D/B - 18 BIT CPU		\$ 4.95
TMS 9901NL MICRO-P PSU		\$ 5.95
TMS 9904NL MICRO-P CLOCK GEN AND DRIVER		\$ 5.95
TMS 9918ANL MICRO-P COLOR GRAPHICS AND DISPLAY		\$ 9.95
KEYBOARD (99/4) 48 KEYS MEASURE 4 x 10 (HETER)		\$ 9.95

APPLE II and APPLE II+ COMPUTER MAINFRAMES (fully populated) \$150

Power supply, case and keyboard, separately available Call or Write

Unit as described above, fully assembled & tested \$250 plus shipping

APPLE POWER SUPPLIES \$29.95

Cassette Software:

I have cassette software send for list (reg \$9.95-\$14.95)

Hal's Special Price: 1 for \$3.95, 3 for \$10.95, 8 for \$19.50, 10 for \$29.50, 20 for \$50.00

or let Hal select 25 different cassettes (my choice) at 25 for \$50.00!!

HAL'S COMPUTER GOODIES

Times printer (reg \$99.95 now \$59.95)

Coleco cassette player (battery op.) \$29.95

9V @ 800 Ma adapter (needed for 18K RAM packs) \$7.95

18K RAM pack module, new \$29.95

18K RAM pack module, refurbished \$19.95

RCA TV interface cable \$1.25

Dual jack cassette interface \$2.50

TV/Computer game switch \$3.95

SHIPPING INFORMATION: ORDERS OVER \$25 WILL BE SHIPPED POST-PAID EXCEPT ON ITEMS WHERE ADDITIONAL CHARGES ARE REQUESTED ON ORDERS LESS THAN \$25 PLEASE INCLUDE ADDITIONAL \$2.50 FOR HANDLING AND MAILING CHARGES MICHIGAN RESIDENTS ADD 4% SALES TAX SEND 20¢ STAMP OR SASE FOR FREE FLYER CANADIAN ORDERS ADD \$5.00 POSTAGE IN U.S. FUNDS

HAL-TRONIX, INC.
P.O. BOX 1101 - DEPT. H
SOUTHGATE, MICH. 48195
PHONE (313) 285-1782

"HAL" HAROLD C. NOWLAND
W82XH

Dear Mr. Frost,

First, there are three capacitors inside the 2040 near the big chip. C6 is tied on D0 to ground. This data line is also the printer ready line. Apparently your printer is ready too fast. Remove this cap and your problem should be fixed. You should remove all three of them.

Second, YES and SI are two numeric variables and "YES" is a literal string. the correct way to develop your program is as follows:

```
10 LET YES$="SI"  
20 PRINT YES$
```

There are some educational programs available for the 2068 and Spectrum. Check with your favorite distributor for titles.--tb

Missing Bytes Dept.

Regarding the article in last issue's FORUM column about the Time Designs/SUM merger, it seems a very vital piece of information was either deleted, forgot, or lost in "MAX", my souped-up TS1000, I was using as a word processor; Time Designs address is TIME DESIGNS, 29722 Hult Road, Colton, Oregon 97017 and has a yearly subscription rate of \$15.00. We hope Tim Woods will accept MAX and I's sincere apology.

Dear Editor,

I am enclosing \$19.95 U.S. for a subscription to SyncWare News (thank you-ed.). Fred Nachbaur and Peter McMullin kept bugging me so now I'm ordering it. I also plan on getting all the back issues (thanks again-ed.).

Even though I have never seen SWN I feel it needs more ZX81 content. But we all know how the 2068's get if their "precious" machines don't receive enough attention.

Sean Wenzel
Toronto, Ontario

Dear Mr. Wenzel,

Judging anything sight unseen seems hardly fair, but everyone is entitled to an opinion. Here's mine:

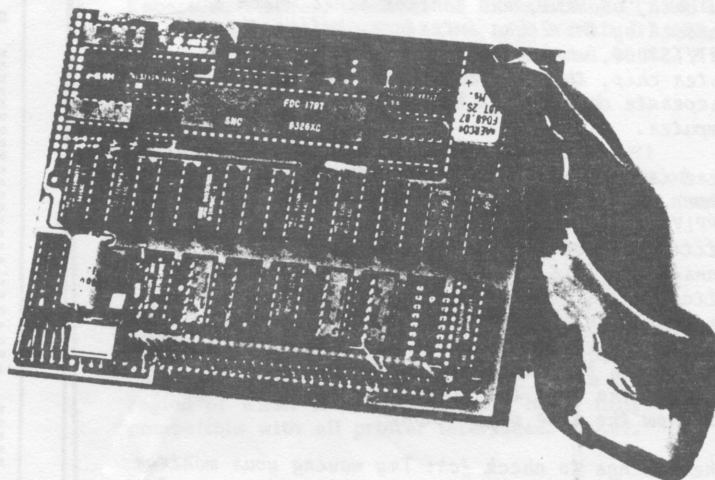
We try very hard to print a balance of articles for ALL the TS computers. Within the limitations of our space, we try to print a software and hardware article each month for both the ZX81/TS1000 and the TS2068. However, if the material isn't there, it's just not there. If you'll pardon the paraphrase, "Sometimes ye have not because ye write not". If no one writes about a specific machine, then support for that machine will die. Human thought is mostly associative. Meaning an idea you have and publish will trigger another idea in someone else, and so on. Many times a simple question asked in this column has lead to a good in-depth article exploring a facet of one of the computers heretofor unexplored.--jm

THE AERCO FLOPPY DISK INTERFACE RUNS 3.2 MEGABYTES OF ON-LINE CP/M SOFTWARE... 50% FASTER THAN KAYPRO.

Don't forget our printer
and serial interfaces too!

AERCO
ACME ELECTRIC ROBOT CO.

BOX 18093 AUSTIN
TEXAS 78760-8093
(512) 451-5874



CP/M is a registered trademark of Digital Research, Incorporated.

UV Lamp Use Warning

Dear Editor,

I saw your update on using old Sol to erase EPROMS on page 6 of SWN #3:5.

The using of an "Ozone" or "Steri-lamp" is okay, but having worked some years in the lamp division of Westinghouse, I must warn your readers that the UV from these lamps can cause more than "spots before your eyes"! The UV at 2537 angstroms not only kills germs, but cells and causes inflammation in the eye by giving it a good burn. The shield used for this type of lamp should be opaque and not mirrored. It was quite common for eye damage to occur from a reflecting surface, even if the lamp was itself concealed. They were used in refrigerators and places not viewed or occupied for long.

The so called "black light" UV from special fluorescent tubes at about 3600 angstroms is not so bad, but I don't know how efficient it is in erasing an eprom (it will work, but takes 4E4 to do it-fn). The fluorescent lamps also require some sort of ballast system to work. By the way, the "spots or blur" from these lamps is due to the eye being unable to focus these wavelengths.

Regarding RFI (page 4), I operate my ham transmitter up to 1 KW while saving to disk and printing. I use my ZX81 to send Morse code without bothering anything yet I have five or six ribbons of up to 4 feet in use. So it is possible to have things work. The only thing I found really bad was any ribbon extension from the rear edge connector of the TS1000/ZX81...that caused havoc in the receiver and was easily spiked by the transmitter...yet a ribbon from the keyboard sockets to an external keyboard causes no problems at all.

Bob Howard
West Covina, CA

New Spectrum Announced

The latest in Britain's best selling Spectrum line of home computers made its debut at the PCW exhibition in Olympia, Great Britain, over the Labor Day weekend.

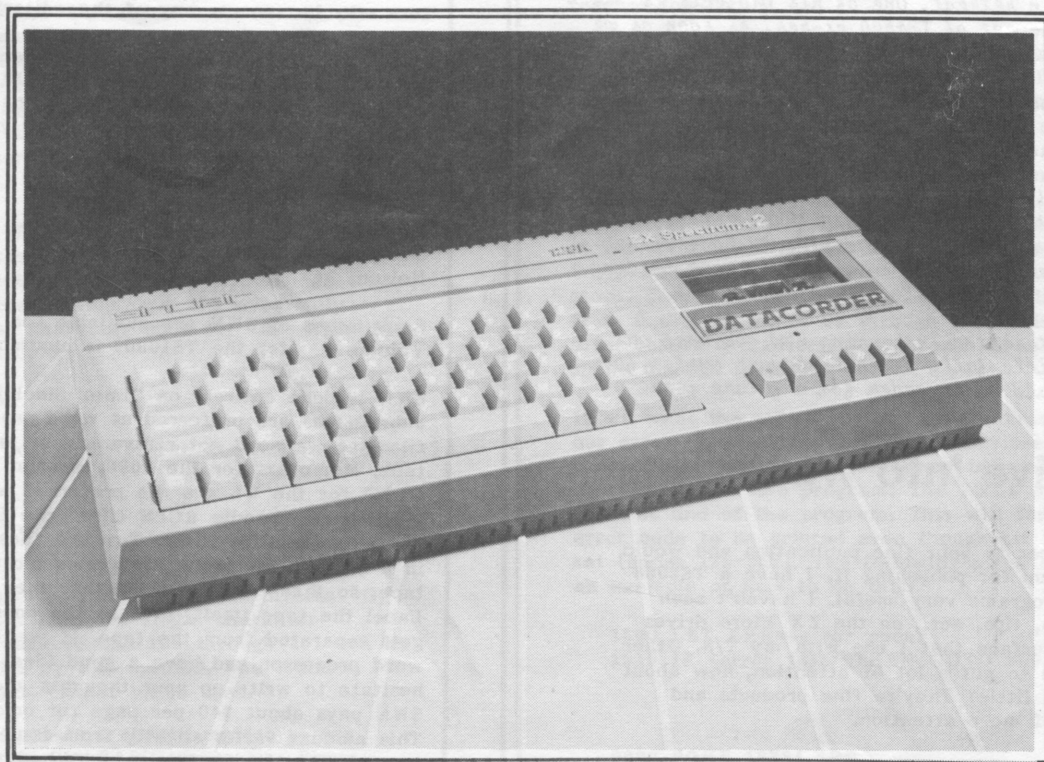
Designated the Sinclair Spectrum 128K + 2, this new machine is the first Sinclair branded computer to be designed and manufactured since the Sinclair brand was acquired by Amstrad Consumer Electronics plc in April this year. However, there are no immediate plans to manufacture or market a version for the United States.

The Spectrum 128K + 2 has an impressive list of features including a real typewriter-style keyboard, built-in datacorder and joystick ports, and full function calculator. The unit is advertised as being ready to accept serial printers, a monitor, keypad, a modem, and the many other peripherals to help with today's computing needs. Provisions have been designed in to allow a music synthesiser to be connected to the MIDI port to allow music to be composed and performed through the Spectrum's four channel high quality sound port.

The Spectrum 128K + 2 is supposed to be compatible with the Interface One and Microdrives, and most other hardware and software items produced for the Spectrum 128.

For more information, contact:

Nick Hewer
Michael Joyce Consultants Ltd.
19 Garrick Street
London WC2E 9BB
Great Britain



Report From Burlingame

Dear Editor,

I was especially interested in the short article on the TS1500. I teach in the Music Department of the local Community College and we have set up a C.A.I. lab to assist students with their music study. This is a project we had hoped to launch for years but were never able to do so due to budgetary limitations, the high cost of Apple Computers, and available software to suit our needs. Now we have more than we could ask for, thanks to six TS1500's modified for monitors, and equipped with polyphonic music boards similar to the MUSIC BOX IV boards from Rave Research (no longer available). The boards have a built-in headphone amplifier so the students work privately, although an accessory power amp can be switched in for group participation. The best part of the whole thing is that we have been able to design our programs to suit our specific needs rather than having to depend on commercial software. We are particularly proud of our lab. It was accomplished at a mere fraction of the cost of comparable commercially available systems.

Anyway, our two limitations are in the areas of program loading time and hi-res graphics for music notation. We thought we had solved the loading question with the possibility of using EPROMs in Rompak boards, but haven't figured out how to make them work on the 1500. The reference to hi-res possibilities in your article (SWN #3:3-ed.) really sparked my interest. I hope more information will appear soon!!!

John Hancock
Burlingame, CA

Dear Mr. Hancock,

Rest assured that the author of the brief article you read in SWN #3:3, our own Fred Nachbaur, is not hibernating under the snow capped peak of his Silicon Mountain retreat. One of his projects is further development of TS1500 hi-res. As soon as we hear of any news we'll let you know.

Looking over the article, I noticed that one of the requirements to produce the hi-res screens on the TS1500 was to have full decoding in the 8-16K region of memory. A Hunter board was listed as one of the ways this could be done. The Hunter board is essentially a Non Volatile Memory (NVM) expansion unit. It would seem that your desire to have hi-res screens could also provide an answer to your loading time problems.

We would be interested in hearing how you make out. Perhaps an article fully describing your system for our readers would be in order for a future issue of SWN.-jm

Microdrive Info Wanted

Dear Editor,

I am a subscriber to your fine publication and would like to thank you for publishing it. I have a TS2068 and find the programs very useful. I haven't seen any information, tips, ect., on the ZX Micro drives and the ZX Interface that I use with my T/S. Other hardware seems to get a lot of attention, how about evening it up a little? They're fine products and are deserving of more attention.

R. L. Pace
No. Hollywood, CA

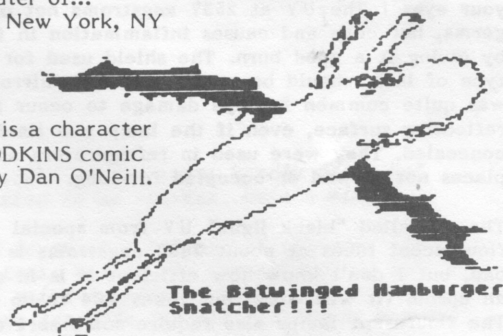
Dear Mr. Pace,

To our knowledge, no one has submitted any microdrive conversions or articles to SWN. It would seem that the majority of American users prefer to go with disk drives. If you, or any other reader, have some good information to share on this subject, we would be glad to publish it as space allows.-jm

From the computer of...

Francine Sklar New York, NY

The "Snatcher" is a character of the ODD BODKINS comic strip created by Dan O'Neill.



The SyncWare News

Thomas B. Woods
Jeffrey D. Moore
Thomas J. Bent
Fred N. Nachbaur

Publisher
Editor
Associate Editor
Technical Director

All subscription information, billing and ad artwork should be addressed to:

SyncWare News
PO Box 64
Jefferson, NH 03583

Phone: (603) 586-7734

Article submissions, Forum, Support listings, Classified ads and any questions you may have concerning the use or everyday abuse of Timex Sinclair computers may be sent to any of the editorial offices.

Jeff Moore
602 S. Mill St.
Louisville, OH 44641
(216) 875-1257

Tom Bent
9016 Flicker Pl.
Columbia, MD 21045
(301) 730-7187

In Canada

Fred Nachbaur
Mountain Station Group Box C-12
Nelson, BC V1L 5P1 Canada

Phone: (604) 354-3858

Back issues of SWN are available for \$3.50 each.
Volume # (for the TS1000) is available for \$16.95

SWN is done entirely on Timex Sinclair computers. Submissions are preferred as word processed text files (so we need not reinvent your article) on good tape. Memotext for the 1000, Mscript for the 2068, QLWP for the QL are the preferred word processors. We can also handle RCPM files done in WordStar (or Ascii format CPM files). Tasword files must be done in a 52 column format. Programs must be submitted on tape, so that we can verify that they will run. Label the tape itself, in case the documentation gets separated from the tape. If you do not have a word processor and have a good idea, then don't hesitate to write up your thoughts and submit them. SWN pays about \$40 per page for original articles. This amount varies slightly from issue to issue and is paid when the article is published. Write for more information.

TS1000 Error Reports: Basic & Advanced

Peter D. Hoffman
Corpus Christi, TX

In this article, we will explore how error reports work on the TS1000/ZX81 computer, and furthermore, how to use "custom" error reports in your own programs.

We are all familiar with the TS error report system, even if all our programs are perfect from the outset (ri-i-ight!). The error report appears at the bottom of the screen, separated by a slash from the line number at which it occurred. If the program runs perfectly, the report 0/line_no. is seen. If there is an error, a report code of 1 to F appears instead.

The error code is kept in memory location 16384 (4000 hex), the first byte in RAM memory, which is also the first system variable. This variable goes by the name ERR_NR. We can check on the value of ERR_NR by typing the immediate command:

```
PRINT PEEK 16384
```

The number 255 will be printed to the screen. 255?? The error report at the bottom is zero! Why 255?

You have just discovered the first characteristic of the error report system. The value in ERR_NR is INCREMENTED before being printed. A value of 255 (FF in hexadecimal notation) is the largest number that a single byte or eight-bit register can hold. If a byte or register holds 255 and is incremented, the value rolls over and becomes zero. This is like an automobile odometer when reaching 99,999.9. Since incrementing FFh results in zero, we can also consider FFh to represent a minus 1. This is the view that applies in this case, and explains the error report of zero and the printed ERR_NR value of 255.

Actually, a more precise explanation is that, after the value is incremented, an additional 28 is added so that the correct character code is produced for printing to the screen. Thus, our 255 eventually ends up as 28, the character code for "0." With this information in hand, try

```
POKE 16384,34
```

as an immediate command, or run the following one-line program:

```
10 POKE 16384,243
```

The results are error codes of "Z" and inverse "G," respectively. In fact, all the single-character codes from 0 through 63 and 128 through 191 can be used as error codes by this method. The multi-character commands cannot be so used.

At this point, it appears a simple matter to use error reports in our own BASIC programs: First, a line to test for the error condition (and branch

around if false, or no error), followed by a line to make the POKE to location 16384. Well, almost but not quite. Run this simple two-line program:

```
10 POKE 16384,143  
20 REM
```

The resulting error report is 0/20. So, what happened to the inverse G? However, if we change line 10 to read POKE 16384,34, the report is Z/10 as expected. What is the difference?

We have just found several more characteristics of the error reporting system. The error report is printed at the bottom of the screen:

- (1) after an immediate command,
- (2) after the last line of a program or STOP, or
- (3) after an error is detected in executing a program line.

As each line is executed, a test is made of the value in ERR_NR to see if an error occurred. The test is whether Bit 7 (the most significant bit) of ERR_NR is set. The test is not, as you might think, whether ERR_NR is 255 (FFh). This means that any value from 128 through 255 will test as being not-an-error. Furthermore, as each line is executed, the value of ERR_NR is reset back to 255.

The above explanation accounts for the fact that the inverse G error report didn't occur in the two-line program. Line 10 does indeed POKE 143 into ERR_NR. The test, however, shows that bit 7 is set, so no error is indicated. As execution of line 20 commences, the value in ERR_NR is set to 255. Thus we get the error report 0/20 when the program reaches the end.

If we are satisfied with a somewhat expanded, but incomplete, set of error reports, then a single line can be inserted at each place in the program where an error report is needed.

```
IF <error_test> THEN POKE 16384,<error_code>
```

The <error_test> is just some expression to evaluate if the error condition has occurred. If the test is true, then the POKE is performed, and the program stops. The value of <error_code> can be in the range from 0 through 34 or 99 through 127. This will give error report characters of 1 through Z and inverse space through inverse 0 (zero).

If we want the entire set of single characters for our error reports, then we must make use of the characteristic that an error report is printed after the last line of the program. The POKE must be in the last line of the program. This will force the error code to be printed even though bit 7 may be set (codes 128-255). The following piece of code is an example of how to do this:

```
1500 LET ERR=<error_code>  
1510 IF <error_test> THEN GOTO 9999  
.  
.  
.  
9999 POKE 16384,ERR
```

The first two lines are inserted everywhere in the program where an error report might be needed. Line 9999 must be the last line in the program. To produce all of the single-character codes by this method, the value of <error_code> can range from 0 through 34, 99 through 162, and 227 through 255.

Now, for all you machine-code aficionados, I will first show and discuss the use of a brief machine-code routine in a BASIC listing that will also give a full range of error codes. From there, it's easy to do the same thing in machine-code only. I'll start with the "how-to," then go on the (more important) explanation of how it works.

The following piece of code will give all the single-character error codes:

```
0010 REM INT x
.
.
1500 IF <error_test> THEN GOTO 1530
1510 POKE 16515,<error_code>
1520 RAND USR 16514
.
.
```

The REM line is set up to reserve two spaces, e.g. 10 REM xx, and then doing a POKE 16514,207 as an immediate command. The three lines of BASIC are inserted anywhere that an error report code is needed. The <error_test> and <error_code> labels have the same meaning as before.

There is a more memory-efficient way of accomplishing the same result, as shown below:

```
0010 REM INT X INT X INT X ....
0020 LET BASE=16514
.
.
1500 IF <error_test1> THEN RAND USR BASE
.
.
IF <error_test2> THEN RAND USR (BASE+2)
```

The REM line is constructed similarly to the first example. Two spaces are reserved for each error report desired, and each even location (16514, 16516 etc.) is POKEd with 207. The odd locations are POKEd with the various error codes needed. Single lines of BASIC are inserted anywhere that an error report is wanted, with the machine-code jump after the THEN going to the selected error routine.

How can a simple two-byte machine-code routine, in which the second byte is used solely as data, produce an error report? The answer lies in the somewhat special properties of a set of Z80 machine-code instructions known as "restarts," and in how the Sinclair ROM designers used these capabilities.

The Z80 instruction set is provided with eight restart instructions. A restart is essentially an unconditional CALL to a specific memory address. The eight restarts are RST 00, RST 08, RST 10, etc. up to RST 38 (in hex numbers). It is up to system designers to determine the use of this capability, which is generally to call often-used routines from many places within the ROM or other operating system. Restarts are one-byte instructions (as opposed to three bytes for CALL), so they make for efficient use of ROM memory locations.

The designers of the ZX81 ROM chose to use the restarts for such routines as "collect a character," "print a character," "collect next character," "turn on floating-point calculator," etc., and of course, "error restart." The error restart is RST 08h, whose hexcode is CFh, i.e. the 207 that we POKEd into the REM line.

An important point to remember is that restarts are calls (like GOSUBs), not jumps (like GOTOs). A call in machine-code has the same meaning as in higher level languages; the location or address from where the call was made is stored, so that a return to the calling routine can be made when the call is complete. Thus, when we do a RAND USR 16514, the first instruction encountered is a RST 08h. This immediately pushes the address of the next byte (i.e. 16515) onto the stack, and then jumps to the error restart routine at address 0008.

When any of the other restart routines are called, the stacked return address is eventually used to get back to the place where the restart was called from. The error restart is the exception. No return is ever made to the calling routine! Rather, the return address is used as a pointer to the data byte that immediately follows the RST 08h instruction (i.e., the desired error code). Furthermore, by going directly to the error routine, there is no test of bit 7, so that any single-character error report can be used.

This is precisely how the ROM does the fifteen error reports that the ZX81 operation system contains. There is simply a RST 08 instruction at many places in the ROM, each followed by the appropriate data byte to produce the error report. This is, of course, preceded by the necessary code to test for the error condition and branch around the restart and data byte if no error is detected.

If you are programming entirely in machine code, it should be obvious by now how to produce error reports; do what the ROM does. Insert your code to test for the error and branch if no error, add a RST 08, and follow it with the code for the desired error report. Remember, however, that you cannot get back to the same point in your routine, since the return address is popped off the stack, used to load the data byte, and destroyed in the process.

Now the good news: The entire stack is cleared (reset), so it doesn't matter how deep the stack is when you call RST 08h. You therefore don't have to bother popping back whatever values you have on the stack before going to the RST 08. The BASIC GOSUB stack is NOT cleared, however, because of the clever way in which the machine stack and the GOSUB stack are controlled by the operating system. A discussion of this is beyond the scope of this article, but if you're interested in experimenting with this see the "Fun with Hot Z" tyd-byt in a previous issue.

Error reports can be useful to the pure machine-code programmer as a means for tracing a program that crashes at some unknown point. Provide two NOPs (00h) at critical junctures in your program. These can then be turned into error traps by poking in a CF followed by data byte. If the program makes it to that point, an error report will be generated and the routine will return to a nice, safe BASIC warm boot, regardless of what might be on the stack.

Redefine the TS2068 Character Set

Kenneth E. Majors

Every 2068 user should know by now that they can define their own graphics characters, but did you know that you can also design your own type fonts? Well now you are no longer locked in by the standard 2068 block characters. The really amazing part is that it is quite easy to redesign them. The system variable CHARS (23606 & 23607) hold the address of the character table minus 256. POKEing this variable will change where the 2068 looks for the characters to print on the screen or 2040 printer. Try this in command mode:

POKE 23606,0: POKE 23607,0

See those strange boxes where the error report should be? Type a few other letters or numbers. Nothing but weird stuff, but kind of fun. What we did was tell the computer that the character table started at address 256 (00FFh), remember that CHARS is 256 less the start of the table. Well that was fun but how do we fix it?

Carefully type, you won't be able to see a mistake, POKE 23607,60. Now everything should look back to normal. If not try again or simply turn the computer off then back on again. You may be wondering how the computer can interpret such things as we had before. The computer doesn't read characters nor does it care what they look like. It prints strictly for the human operator. It reads the keys that the operator/programmer presses and prints a series of dots so you can see what key you pressed. This is where the fun comes in. Because of this we can tell the computer where we want to look for that dot table.

Enter the program in figure 1 and press RUN. It takes a few seconds for it to get ready for you, but be patient. When the screen appears press 1 and enter. You will be asked which character you want to view, press the space bar and enter. The screen clears and an empty 8X8 grid is drawn. Not too impressive yet.

figure 1

```

1 CLEAR 61652: FOR X=1 TO 776
: POKE (64758+X),PEEK (15615+X):
: POKE (61652+X),PEEK (15615+X):
NEXT X
2 POKE 23606,247: POKE 23607,
251
3 GO SUB 5: GO TO 17
5 DIM a$(8,8)
6 FOR X=1 TO 8: LET a$(X)=""00
000000": NEXT X
10 FOR X=88 TO 152 STEP 8: PLO
T X,159: DRAW 0,-64: NEXT X
11 FOR X=159 TO 95 STEP -8: PL
OT 88,X: DRAW 64,0: NEXT X: RETU
RN
17 INPUT "1=view a char      2=
modify char"i$
18 IF i$="1" THEN INPUT "enter
character"i$: GO TO 1000
20 LET z$="X": LET x=2: LET y=
11: LET z=0
25 PRINT AT 12,0:"ss-a= char f
inished":TAB 0:"ss-s= clear":TAB
0:"ss-f= save":TAB 0:"ss-g= los
d":TAB 0:"ss-y= quit":TAB 0:"1=
0=space":TAB 0:"5,6,7,8 = m
ove cursor"

```

```

26 PRINT AT 11,13:"10X",AT 11,
26:"1X"
40 PRINT OVER 1:AT X,Y:Z$: OVE
R 0
45 LET a=X: LET b=Y
50 LET c$=INKEY$: IF c$="" THE
N GO TO 40
60 LET y=y+(c$="8")-(c$="5")
70 LET x=x+(c$="6")-(c$="7")
80 LET x=x-(x>9)+(x<2)
90 LET y=y-(y>18)+(y<11)
95 PRINT AT a,b:("■" AND a$(a-
1,b-10)="1")("■" AND a$(a-1,b-1
0)="0"): PRINT OVER 1:AT X,Y:Z$
OVER 0: PRINT AT 8,26:""
100 IF c$="1" THEN PRINT OVER 0
:AT X,Y:"■": LET a$(x-1,y-10)="1
110 IF c$="0" THEN PRINT OVER 0
:AT X,Y:"": LET a$(x-1,y-10)="0
113 FOR a=8 TO 1 STEP -1: FOR c
=1 TO 8: IF a$(a,c)="1" THEN PLO
T 207+c,112-a
114 NEXT c: NEXT a
115 IF c$="" STOP " THEN GO SUB
2000: PAUSE 10: CLS: GO SUB 6:
GO TO 17
116 IF c$="" TO " THEN GO TO 300
0
117 IF c$="NOT " THEN CLS: GO
SUB 5: GO TO 20
118 IF c$="" THEN " THEN GO TO 3
040
119 IF c$="" AND " THEN STOP
120 GO TO 40
1000 FOR a=1 TO 8
1010 LET d=PEEK (255+(PEEK 23606
+256*PEEK 23607)+(CODE i$-32)*8
)+a)
1020 FOR c=8 TO 1 STEP -1
1030 LET a$(a,c)=STR$ (0+d-2+INT
(d/2)): LET d=INT (d/2)
1040 NEXT c: NEXT a
1050 FOR a=1 TO 8: FOR c=1 TO 8:
PRINT AT a+1,c+10:("■" AND a$(a
,c)="1")("■" AND a$(a,c)="0"):
NEXT c: NEXT a: GO SUB 10: GO TO
20
2000 INPUT "What character to as
sign?"i$
2010 FOR X=1 TO 8: GO SUB 2050
POKE (255+(PEEK 23606+256*PEEK 2
3607)+(CODE i$-32)*8)+X):d: NEX
T X: RETURN
2050 LET d=0: FOR c=1 TO 8: LET
d=d+2*VAL a$(X,c): NEXT c: PRINT
AT X+1,0:d: RETURN
3000 INPUT "name of font?"x$
3010 SAVE x$CODE (256+(PEEK 2360
6+256*PEEK 23607)):776: GO TO 30
3040 CLS: POKE 23606,213: POKE
23607,239: PRINT AT 8,0:"1= Norm
al"
3041 POKE 23606,0: POKE 23607,60
3041 PRINT "5= User designed (pr
evious save)": PRINT "6= Return
to User area"
3045 INPUT "Select choice"i$
3046 IF c$="1" THEN CLS: POKE 2
3606,213: POKE 23607,239: GO TO
3
3050 IF c$="5" THEN CLS: GO TO
3050
3051 IF c$="6" THEN CLS: GO TO
2
3052 GO TO 3045
3055 INPUT "name of font?"x$
3060 LOAD x$CODE 64759,776: GO T
O 2
5000 REM CHARACTER SET BUILDER
BY K. E. MAJORS

```

Draw a box around the outside edges with the unshifted cursor control keys and the 1 key to print a dark spot or the 0 key for a light spot. When you finish press Symbol Shift-a (STOP). You are asked which character to assign, press the space bar again and enter. A line of numbers will come down the left side of the grid, more about them later. The screen will clear and the begining prompt is printed again

except there is a box every where a space was before. Now that's a little more impressive isn't it?

To put it back to normal just do the same thing again and either erase the box with the 0 key or press Symbol Shift-s (NOT) to clear the grid. Then reassign the space as before.

Every printable character can be redone in this manner. The double characters, >=, <>, and <= can not be done because they are assembled the same way that keywords are when they are used. To change them you have to change all the elements that make them, both the > and the =. Any font style that the user can create on an 8X8 matrix can be done with this routine. They can be SAVED (Symbol Shift-f) or LOADED (Symbol Shift-g) and are fully printable on the 2040 printer.

Many larger printers can accept downloaded characters from the computer. Try defining a few more. The character is drawn in normal size just to the right of the large grid so you can see what it will look like when it is printed.

How it Works

First let's look at how the characters are made. It takes 8 bytes to store 1 character. This table is displayed by the computer in binary format. A 1 in the binary number will print an ink spot and a 0 will print a paper dot. As an example let's use the "!" character:

```

HEX BIANARY CHARACTER
00 00000000 : : : : : : : :
10 00010000 : : : : X : : : :
10 00010000 : : : : X : : : :
10 00010000 : : : : X : : : :
10 00010000 : : : : X : : : :
00 00000000 : : : : : : : :
10 00010000 : : : : X : : : :
00 00000000 : : : : : : : :

```

Can you see how the "!" is formed? All printable characters are formed in the same way. If we change the positions of the ones in the pattern above, the shape of the "!" is altered accordingly. Since we can change the place where the computer looks for the table, our imagination and 8 bytes per character, is our only limit on type fonts.

The Program

The first thing this program does is move RAMTOP down to give us some "protected" space. It takes 776 bytes to hold a complete character table. The ROM table, located at 15616 (3D00h) is copied to the reserved area. CHARS is POKEd to point to this area so we can have immediate results on all our changes. This can be dangerous if you have all zeros in the grid and assign it to something besides a space because you won't be able to see that character when it's key is pressed.

A\$ is DIMensioned to give an 8X8 matrix then is filled with zeros. The grid is drawn and the first input prompt is displayed.

The subroutine at line 1000 finds the requested character and loads it into A\$ in binary form then PRINTs it on the grid. The formula in line 1010 is how the dot pattern is looked up. See if you can follow it. A FOR-NEXT loop is assigned to A.

```

LET D= PEEK (255+( PEEK 23606+256* PEEK
23607)+((CHR$ i$-32)*8)+A

```

First the address of the table minus 256 is found by PEEKing CHARS (23606 & 23607). 32 is subtracted from the CODE of the character in i\$, a space is CHR\$ 32 and the start of the table. The result is then multiplied by 8, 8 bytes for each character remember. Next 255 plus the value of A is added, remember CHARS is 256 less than the starting address of the table, the result is the starting address of the desired characters dot pattern.

Let's walk through one. A space is the easiest to understand. We know that our table starts at address 64759 so CHARS holds 64503, 256 less than start of our table. The code of a space is 32 so our result of CODE i\$-32 is 0. 0*8=0, so when we add results so far we have 64503. The next part adds 255 to 64503, the result of the first two parts to give us 64758 plus the value of A, which this time is 1 on the first pass, and we have 64759, the start of our character table. Try this a few times with other characters until you are comfortable with it. It just looks more complex than it is.

Lines 1020 through convert the hex number into a binary form and loads it into A\$. This is done by subtracting 2 times the INTEger of D divided by 2. the result will be either a 1 or a 0. Try it and see for yourself how it works.

The subroutine starting at line 2000 assigns the newly designed character to the proper place in the character table. This spot is found the same way as before. The binary number in A\$ must be converted to a decimal number so it can be POKEd back into our table. Line 2050 accomplishes this by the multiply and add method. D is set to 0 to start and then multiplied by 2 and the value of A\$(x,c) is added until the final number is obtained.

Lines 3000 through 3010 save the font style for future use in your own programs. It can be loaded anywhere you have 776 bytes of unused RAM. Just load it and poke CHARS with 256 less than the starting address of the table.

Lines 3040 to the end LOAD a previously saved font style.

Applications

The ability to completely redesign the character set lends itself to a host of possibilities. An "A" can be assigned to to an "a" to create an all capital set or a totally different font style can be assigned to the lower case letters and both are available from the keyboard instantly. The keyboard can be redefined from the familiar "qwerty" to "abcdef" or any style that is more suitable for you. The keys will still be the same but will print different characters. When CHARS is restored to it's original value your message will be unintelligible until the proper character table is loaded and CHARS is POKEd to the proper value. Characters can be designed lying down for spreadsheets wider than 32 columns on the 2040 printer. More than one font style can be loaded and switched in and out for fancy printing, increased readability, special effects, underlining, bold type, and multi-font printing. This is just the start of the many things that you can do. Let me know what other ideas and applications you come up with.

Edit or add the lines in figure 2 to the program in figure 1. Add the data statements in figures 3, 4, and 5. This part takes a while so don't get impatient. Pay close attention when entering them because it is difficult to edit them later.

Lines 3040 and 3041 show how to switch between the different fonts to do multi-font printing.

Have fun with this.

```

1 CLEAR 61652: FOR x=1 TO 776
2 POKE (64753+x),PEEK (15615+x):
3 POKE (61652+x),PEEK (15615+x):
4 NEXT x: RESTORE : FOR x=1 TO 776
5 READ a: POKE (63982+x),a: NEXT
6 x: RESTORE 6100: FOR x=1 TO 776
7 READ a: POKE (63206+x),a: NEXT
8 x: RESTORE 6200: FOR x=1 TO 776
9 READ a: POKE (62429+x),a: NEXT
10 x
11 3040 CLS : POKE 23606,213: POKE
12 23607,239: PRINT AT 5,0:"1= Norm
13 a": PRINT "2= ": POKE 23606,23
14 a: POKE 23607,248: PRINT "TECH":
15 POKE 23606,0: POKE 23607,60: PR
16 INT "3= ": POKE 23606,231: POKE
17 23607,245: PRINT "STENCIL": POK
18 E 23606,0: POKE 23607,60
19 3041 PRINT "4= ": POKE 23606,22
20 b: POKE 23607,242: PRINT "RSUP":
21 POKE 23606,0: POKE 23607,60: PR
22 INT "5= User designed (previous
23 save)": PRINT "5= Return to User
24 area"
25 3047 IF C$="2" THEN CLS : POKE 2
26 3606,238: POKE 23607,248: GO TO
27 3033
28 3048 IF C$="3" THEN CLS : POKE 2
29 3606,231: POKE 23607,245: GO TO
30 3033
31 3049 IF C$="4" THEN CLS : POKE 2
32 3606,222: POKE 23607,242: GO TO
33 3033

```

figure 2

The image shows a computer terminal screen displaying a hex dump. The screen is divided into three columns: the first column shows hexadecimal values, the second column shows the corresponding ASCII characters, and the third column shows the memory address. The data appears to be a mix of random hex values and some recognizable ASCII text like 'DATA' and 'TECH LETTER CODES'.

[illegible][illegible]

figure 3

reg.
\$69.95

for ALL T/S
computers

IN/OUT PORTS

SALE! \$39.95

Experiment with computer sensing and control

The ideal port for sending signals from the outside world into your computer. Or use it to turn external devices on or off under computer control. 8 input lines, 8 outputs, plus 2 handshake strobes. Solderless female connectors. Now available at tremendous savings!

Thomas B. Woods
P.O. Box 64, Jefferson, NH 03583

From the computer of...
Carl Jones, Medford, MA



BASIL'S COMPENDIUM

Basil Wentworth

This chapter will show you how to add and subtract, and how to "increment" and "decrement" registers; it will tell you about overflow and carrying; and it will introduce the concept of flags.

But before we go on, I remember that I left you last time with a computer full of mega-REM, with the promise that I'd show you what to do with it. OK, here you are.

First generate, as line 2, a mega-REM of exactly 704 nulls, plus the " at the end. You can count the number of nulls by PRINT USR 16517. Next, key in another line:

```
9 STOP
```

and leave that line there for the rest of your life, to protect the top of the mega-REM. Now edit line 2 so that it reads

```
2 PRINT ".....(etc)
```

It will take a very long time for such a long line to come down into the EDIT position, but have patience. Check the length of the mega-REM once more by RUNNING the program. You should get a screen full of dots, with a report code of 9/9, showing that the program hit the STOP at line 9. If you have miscounted, and the screen is not full, you can add extra dots to eke out the display. If the report code is 5/2, indicating that the screen has overflowed, then you have too many dots, and should remove a few. Any other report code means that you are in trouble, and better start over. But let's assume that it came out right.

```
16514 42 E LD HL,
16515 12 E (16396)
16516 64 RND
16517 35 7 INC HL
16518 17 ) LD DE,
16519 156 16540
16520 64 RND
16521 62 Y LD A,
16522 22 - 22
16523 1 = LD BC,
16524 32 4 32
16525 0
16526 237 GOSUB LDIR
16527 176
16528 35 7 INC HL
16529 61 X DEC A
16530 32 4 JR NZ,
16531 247 RUN -9
16532 201 TAN RET
```

FIG 9-1. SCREEN STORAGE PROGRAM

Now use your loading technique to build a 1 REM line like that shown in Figure 9-1, and you're ready to go. Put in a program like

```
1000 PRINT AT 10,3;"BASIL LOVES JOCELYN"
1010 RAND USR 16514
```

and GOTO 1000. You should find that your words are immortalized in line 2, and that RUNNING the program will bring them back to the screen just as

you had them. Any other display can be frozen in the same way, with a couple of limitations:

1. The display may not contain any quotation marks. Even though you carefully use the required "", the computer will store it as a ", with the result that any attempt to PRINT will result in a crash.

2. The USR 16514 command has to be a part of the program that generates the display. If you generate the display separately, and then try to USR it (if you'll pardon my so cavalierly using USR as a verb), the screen will be cleared when you enter the USR command, and you'll find that you have saved a screen full of nothing. A perfectly symmetrical display, but not a particularly useful one.

NOTE: It's possible to use a machine code routine to transfer the REM statement into the PRINT area of the computer (essentially a reverse of what you did to put it into the REM statement). However, this is what I call sending a man to do a boy's job.

Addition

And now back to lessons. Now that we can put numbers into the computer, move things around inside, and persuade it to tell us the value of BC, let's see if we can find out how to make it do something "useful".

Like addition, for instance.

Register A and register pair HL are the workhorses of the Z80 chip. You'll find that they can do many things that the other registers can't cope with. In fact, these registers are so important that they have been given "significant" names. "HL", as you remember, stands for "high-low". And "A" stands for "accumulator".

"Accumulator" is an unfortunately ambiguous word. Besides being a bear to spell, it means many things to many people. To the British, for instance, the accumulator is a storage battery. To the physicist, it can be a device that stores an electric charge. To us, it means something that accumulates data. But all the other registers accumulate data, too. They just weren't around when this one was named.

I think it's safer to use the term "register A", and to remember that this register will do things that some others can't. But be prepared to recognize the term "accumulator" when you see it.

```
62 LD A,
2 2
198 ADD A,
2 2
201 RET
```

FIGURE 9-2. ADDING WITH REGISTER A

	A	B	C	D	E	H	L	N	(HL)
ADD TO A (ADD A,)	135	128	129	130	131	132	133	198	134
ADD WITH CARRY TO A (ADC A,)	143	136	137	138	139	140	141	206	142
SUBTRACT FROM A (SUB A,)	151	144	145	146	147	148	149	214	150
SUBTRACT WITH CARRY FROM A (SBC A,)	159	152	153	154	155	156	157	222	158
COMPARE WITH A (CP A,)	191	184	185	186	187	188	189	254	190

TABLE 9-1. INSTRUCTIONS FOR REGISTER A

The instructions for adding to and subtracting from A and HL are given in Tables 9-1 and 9-2. For the time being, don't worry about the other instructions and registers that you see mentioned there.

Let's start in with the classic addition problem of 2+2. Try the routine given in Figure 9-2. Put the routine into the computer, and PRINT USR 16514. What do you get?

```

62 LD A,
200 2
198 ADD A,
196 2
70 LD C,A
00 LD B,
00 0
201 RET

```

FIGURE 9-3. ADDING WITH REGISTER A (CORRECTED)

Well, what did you expect? Register A has (probably) arrived at the correct value, but you told the computer to give you the present contents of register pair BC. (If you caught this one on the way by, congratulations!) We'll have to shift the data from A to BC before we can look at it.

How? LET C=A, and LET B=0. (Why not LET B=A and LET C=0?) So our complete program is as shown in Figure 9-3.

Try it. Do you get 4?

Overflow

Now, just for fun, try a number that's too big for the register to handle. Use the program in Figure 9-3 to add 2 to 255. Remember that 255 is the biggest number that the A register (or any single byte anywhere) can handle. And the sum that we've set for it comes to 257.

Well, what is the logical thing for the computer to do? It could give up, of course. Or it could start over--it could say 0 for 256, 1 for 257, and so on. RUN the program, and verify that this is indeed what it does. Try adding 255 to other numbers. H'mm--it begins to look as if adding 255 is the same as subtracting 1. So adding 254 would be like subtracting 2, right? Try it and see. Tuck this fact away in your mind--we'll find a use for it later.

Now how about 2-register numbers? Let's try the same sequence, using the HL register. First, 2+2 (Figure 9-4). Print USR 16514. There's that 4 that you expected. Try adding 2+255, and see if you can

	BC	DE	HL	SP
ADD HL,	9	25	41	57
ADC HL,	*74	*90	*106	*122
SBC HL,	*66	*82	*98	*114

* INDICATES PRECEDED BY 237

TABLE 9-2. INSTRUCTIONS FOR REGISTER HL

get 257 that way. Sure, the two bytes together handle this problem easily.

Let's see what happens when you exceed the capacity of the pair (which is 65535, you will remember). Load the HL register with 65535 (255 in one byte, 255 in the other). Add 1 to it. Do you get 0 again? Now add 65535 to 4, and see if you get 3.

Now subtract. Let's go through a similar set of exercises. Follow Figure 9-5. There's 4-2=2. Try other combinations, like 5-2, 6-4, 8-3. Try 5-6. And there's that 255 again, standing in for -1. Subtract 255 from 6. As you would expect, you get 7. The 255 still thinks it's a -1. Hold onto that thought. It'll come in handy when we get to the chapter on GOTO.

```

33 LD HL,
20 2
17 LD DE, **
20 2
25 ADD HL,DE
68 LD B,H
77 LD C,L
201 RET

```

* TWO BYTES OF DATA NEEDED; LSN FIRST

** THERE IS NO DIRECT WAY OF ADDING A NUMBER TO HL

FIGURE 9-4. ADDING WITH REGISTER PAIR HL

And for double-register numbers. Try the sequence in Figure 9-6.

HEY! There isn't any instruction for subtracting a two-byte number. But there's something that looks a little like it--Subtract With Carry (SBC). Try it. Notice that it takes a two-byte instruction: 237 followed by 82. You'll find several of these two-byters, with 237 or 203 as the first of the two

```

62 LD A,
4 4
214 SUB A,
2 2
79 LD C,A
6 LD B,
0 0
201 RET

```

FIGURE 9-5. SUBTRACTING WITH REGISTER A

bytes. (And a whole couple of 'nother sets beginning with 221 and 253.) Think of the first of

the two bytes as a sort of SHIFT key, that gives a new meaning to the following byte.

```

33  LD HL,
4   4
0   0
17  LD DE,
2   2
0   0
-   SUB HL,DE

```

FIGURE 9-6. SUBTRACTING WITH REGISTER PAIR HL

We won't be using the 203 series for a long time. That little giant is way beyond the scope of our present knowledge. And the 221 and 253 won't come along for some time, either. But we're going to call on members of the 237 set again and again.

Let's continue with the problem given in Figure 9-7. So 4-2 is still 2. Try 4-5, and see whether you get 65535, which is the 256-imal way of saying -1.

But what does this Subtract With Carry mean?

```

237  SBC HL,DE
82
66  LD B,H
77  LD C,L
201  RET

```

FIGURE 9-7. COMPLETION OF FIG. 9-6

Carry

Use the program of Figure 9-7 once more to prove that 4-2 is still equal to 2. Now enter the following bytes at the beginning of the program, before the 33. As always, be sure that you have enough nulls in the 1 REM statement to accommodate the whole program.

```

62  LD A,
2   2
214 SUB A,
3   3

```

Use PRINT USR 16514. What's this? Does 4-2=1 all of a sudden? Try PRINT USR 16518 instead of 16514, to avoid those first four bytes. Now it comes out 2, just as it should.

We learn a number of lessons from this exercise:

1. The main lesson is that SBC (Subtract With Carry) takes into account the recent history of the computer. If you'll notice, those first four bytes have the effect of subtracting 3 from 2, which leaves a negative answer. To let us know that the result of that operation is outside the computer's normal range, the Z80 has "SET a carry flag", as the experts like to put it. Properly speaking, I suppose that the "carry" should be called a "borrow" in this case, but the same flag would have been SET if the first operation had been, say, LD A,255; ADD A,3; giving a true carry. (Try it.)

The carry flag is a perishable signal--if you don't use the information right after the flag is SET, you may be too late. You may find that a subsequent computer operation has RESET it (canceled it) in the meantime.

Add With Carry (ADC) works very much the same way. If the Carry flag is SET, then ADC will add an extra 1 to the sum it generates. Again, try it out. And try ADC and SBC with register HL. Remember, you have to load HL with two bytes of data, LSN first, and you have to LD B,H and LD C,L in order to read out the contents of HL.

It doesn't matter a whit whether or not the action that SET the Carry flag was relevant to the rest of the program. How is the computer to know what is relevant and what isn't? Considerations like that are the responsibility of you, the programmer. All the computer cares is whether the flag is SET--never mind how it got that way.

To sum up, if the Carry flag is SET, a Subtract With Carry (SBC) will subtract 1 from the result, while an Add With Carry (ADC) will add 1 to the sum. If the instruction is Add or Subtract (without carry), then the condition of the Carry flag doesn't matter. And, of course, the computer will then SET or RESET the Carry flag, according to the results of the operation. And again, to reflect the next operation. And again...and again...

The incidental lessons from this exercise are:

2. You can drop in on a program in the middle, as long as the entry point makes sense. If it doesn't make sense, then the results will be meaningless--if indeed the program doesn't crash. You'll notice that when we started the program at 16518, the computer ignored those first four bytes, and zipped through the program as if they didn't exist.

This suggests some interesting possibilities to the adventurous mind. Like--why not put two or three (or more) programs into the same REM statement, and let your USR number signal which segment you want at any given moment? No sweat. You can even arrange it so several programs share parts of the machine code. In fact, this is what the ROM in your computer does.

Or why not start the code with an operation that will be performed in some cases, but not in others? And rely on your BASIC program to enter the machine code routine at the proper point. Sure, you can do that. In fact, you just did.

Or--really wild--why not include some plain language in the REM line, as sort of a REM in itself? Like starting your REM statement:

```
1 REM LINE RENUMBER.. (followed by the machine code CHR$s)
```

Your USR number, then, would be chosen so that the computer skips the plain language and starts at the beginning of the machine code. Or you could put the plain language at the end of the machine code sequence. In either case, you could tell at a glance what this particular segment of machine code was designed to do.

3. The other lesson is that, once you return to BASIC, the program doesn't remember what it did the last time you ran it. It so happens that this program left the Carry flag RESET, but the computer won't remember that next time. Nor will it remember what numbers were stored in the various registers.

The condition of the Carry flag is stored in another register that I haven't told you about yet--the F register. F for Flags. We won't be using

this register as a register, except in a very limited sense; but we will use, time and again, the information that it stores. Other flags that are recorded in this register are Zero (was the result of the last operation zero?), Minus (was the result less than zero?), and a number of others that we won't get around to using for many moons.

```

1 LD BC,
3 3
0
3 INC BC
201 RET

```

FIGURE 9-8. INCREMENT

INC & DEC

There are two special cases of addition and subtraction. One of them, INCREMENT (mnemonic: INC), increases the contents of the designated register by 1. You can see how it works with the program of Figure 9-8. The output should be $3+1=4$.

```

1 LD BC,
3 3
11 DEC BC
201 RET

```

FIGURE 9-9. DECREMENT

	INC	DEC
A	60	61
B	4	5
C	12	13
D	20	21
E	28	29
H	36	37
L	44	45
BC	3	11
DE	19	27
HL	35	43
SP	51	59
(HL)	52	53

TABLE 9-3. INC AND DEC

The opposite of INCREMENT is--no, not excrement, but DECREMENT (mnemonic: DEC). Use the program of Figure 9-9 to demonstrate that 3 decreased by 1 is 2.

The instructions for INC and DEC are given in Table 9-3.

And that's the end of a rather lengthy lesson. There will be more.

~~~~~ DUNGEON OF YMIR ~~~~~

A MULTI-LEVEL MAZE ADVENTURE GAME by Fred Nachbaur (C)1986

FOR THE TIMEX TS1500

FINALLY! A FULL-FEATURE, HIGH RESOLUTION DUNGEON GAME FOR THE TS1500!

This 24K game, written entirely in machine-code, is the most spectacular program ever written for the TS1500. Nine levels, 16 types of monsters, 14 objects, six spells. Easy to play, difficult to master. Includes FAST-SAVE with auto-boot to save in-progress games; time to load entire program reduced to 70 seconds! Revolutionary TRUE HI-RES puts your TS1500 on a par with much larger machines.

Send \$24.95 (cheque or MO) to FRED NACHBAUR, C-12 MTN. STN. GROUP BOX, NELSON BC V1L 5P1 CANADA. Specify version: V1 (TS1500 + 8K Hunter NUM or equivalent) or V2 (TS1500 + 16K RAM pack). V2 requires a minor hardware addition (included). COMING SOON: V3 for ZX81/TS1000. Inquire. *** ALSO AVAILABLE: TS1500 HI*RES EXTENDED BASIC (\$16.95)

~~~~~ DUNGEON OF YMIR ~~~~~  
A MULTI-LEVEL MAZE ADVENTURE GAME by Fred Nachbaur (C)1986

-----THE CAST OF CHARACTERS-----

- \* - This is YOU, the Hero in this saga. You must find
- \* - THE SWORD OF ASHLA, the object of your sacred quest.
- \* - THE ORACLE; perhaps he'll help you, perhaps not....

Along your way, you will encounter many strange things--

~~~~~ MONSTERS ~~~~~

- * - MINOTAUR OF CASTLECAR
- * - ARCHER OF ARGENTA
- * - BIRDMAN OF INVERHERE
- * - DIRE WOLF OF SILVERKING
- * - THREE-LEGGED GREENLID
- * - GIANT KILLER COCKROACH
- * - HORRIFIC BATTLE-BAT
- * - GRISTLY GORY GHOUL

~~~~~ OBJECTS ~~~~~

- \* - MAGICAL SPELL VIAL
- \* - PSYCHIC LAMPLIGHT
- \* - CHEST OF MYSTERY
- \* - STAIRS UP
- \* - SACRED TEMPLE OF ASYLUM
- \* - GOLD TEMPLE OFFERING
- \* - PIT OF CEILING HOLE
- \* - 100 GOLD PIECES

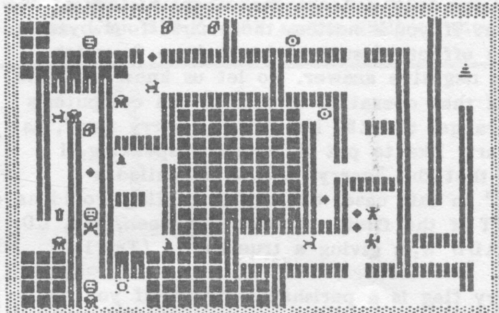
~~~~~ THE CONTROLS ~~~~~

- S - Move through maze
- D - Cast a Drift Spell
- S - Cast a Shield Spell
- H - Take a Healing Potion
- T - Cast a Teleport Spell
- R - Invoke Rejuvenation

~~~~~ REMARKS ~~~~~

BEWARE: If a monster attacks, you must fight to the DEATH!

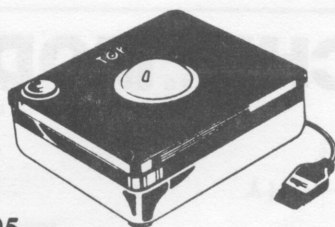
ALL  
ARTWORK  
IN THIS AD  
FROM ACTUAL  
SCREEN  
DUMPS!



A Trapped Gremling  
Hit= 0 Exp= 303160 0 Tel 1 Sh4 Dr0 R11 Heal0



## TS2068 Trackball Only \$19.95



Originally sold for \$69.95

Specify Cat# TBTMX02

**Plugs into TS2068 Joystick Port and works with all joystick software.**

**Bonus Feature:** Also works on Commodore 64, VIC-20, ATARI 800, and more. Contact factory for more complete list.

You can benefit from our recent purchase of brand new WICO Trackball Controllers at closeout prices. We've taken the model WICO originally made for the Texas Instrument 99/4A and made a very simple modification so it now is fully compatible with the Timex TS2068's joystick port.

WICO is the largest designer and manufacturer of control devices for commercial arcade video games. If you've ever played an arcade video game, chances are you've used a WICO joystick or trackball. You've experienced the superior control. The pinpoint firing accuracy. The exceptional durability.

**Features:** Phenolic ball offers 360-degree movement. Two optical encoders provide split-second movement. Quick-action fire button for smooth, two handed arcade response and feel. Long 5' computer connection. Heavy duty plastic case for long hard use.

The WICO warranty has been voided by our modification. But we give you our 15-day money back guarantee and a one-year limited warranty from Zebra Systems.

## Timex Games \$2 Each

With your order for a TS2068 trackball you can purchase any of the following Timex TS2068 Trackball and Joystick compatible games at the special low price of \$2.00 each for cassettes and \$3.00 for cartridges.

CAT# TITLE

Cassettes at \$2.00 each

64001 Androids  
64002 Penetrator  
64004 Casino I  
64005 Crossfire  
64006 Circuit Board Scramble  
64007 Dragmaster  
64009 Guardian  
64012 Fun Golf

CAT# TITLE

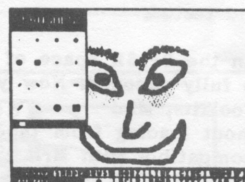
64014 Hungry Horace  
64015 Horace Goes Skiing  
64019 Horace and the Spiders  
64021 Blind Alley  
64023 Crazybugs

Cartridges at \$3.00 each

74001 Androids  
74005 Crazybugs

## \$5 Off Tech-Draw Jr.

You can save \$5.00 on the purchase of Tech-Draw Jr. if you purchase it at the same time as a TS2068 trackball. Instead of the regular price of 19.95 you can get it for 14.95. See our catalog for a complete description of Tech-Draw Jr. and a list of printers that it supports. Order Tech-draw Jr. Catalog# C256.



## TS1000 TRACKBALL Only \$39.95

Originally sold for \$109.95

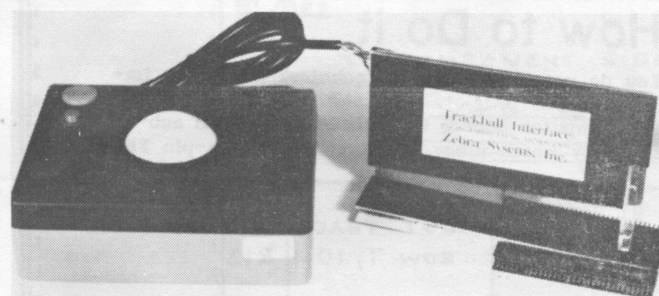
Specify Cat# TBTMX01

**Plugs into the back of  
TS1000,1500,2068, or ZX81.**

We've taken WICO's Apple II trackball and put its controller card on an interface adapter for the Timex bus. Now you can get all the benefits of the Apple Trackball with its intelligent controller card, on your Timex computer.

The Apple trackball controller has sixteen integrated circuits on it that read the optical electronic encoder wheels from the trackball and completely keep track of the trackball movement with separate x and y direction up/down counters. This enables your Timex computer to get the position of the trackball by just reading two input ports. This is a tremendous advantage on the TS1000, since the computer can be left in "SLOW" mode for smooth graphics, while the trackball interface card does all the work.

The Apple II trackballs alone originally sold for over one hundred dollars. Now you can take advantage of Zebra's recent purchase of a large number of them at closeout prices. You get the Apple II trackball with Apple interface card, Zebra's Timex- to-Apple bus adapter, and complete instruction manual with sample routines for all the Timex computers. And all for just \$39.95.



**Ordering Instructions:** Include \$3.00 S&H for UPS. P.O Boxes and other orders requiring U.S. Mail must add \$4.00 extra shipping per trackball. VISA/MC Accepted. NY Residents add sales tax. Order now! Quantities are limited to stock on hand.

## Zebra Systems, Inc.

78-06 Jamaica Ave.  
Woodhaven, NY 11421  
(718) 296-2385  
HOURS: M-F 9AM-5PM

### TRACKBALLS FOR OTHER COMPUTERS

We have bargain priced WICO trackballs adapted to just about every popular computer on the market. Send a SASE or call for a complete list.

# 1000 ONE CHIP MOD

## A Built-in NVM

By Gerd Breunung

[EDITOR'S NOTE: We have received several submissions for non-volatile memories based on the 6264-LP 8K static RAM. Though all have their merit, this one is the most elegant. It is fully decoded, yet still requires only one IC (the 6264-LP SRAM). Furthermore, it is installed on the ZX81/TS1000 board itself, and therefore does not require edge connectors, etc. Lastly, all parts except the RAM chip are available at your nearby Radio Shack. Some sources of the 6264-LP are Microprocessors Unlimited, Active Electronics, and Jameco. Check the ads in "Byte" and "Computer Shopper" for other sources.]

This battery backed-up RAM is a miniaturization and functional equivalent of the famous "Hunter" board. It was designed by Mr. Wilf Rigger of the Vancouver, BC T/S User Group. I owe many thanks to Mr. Rigger for inspiring me to write this article, and assisting with technical advice during the construction and refinement of this project. Yes, everything in this article has been built and tested.

This non-volatile RAM resides in the 8-16K space of the ZX/TS memory map, and is fully decoded. Now you can run utilities like Q-Save, Toolkits, Mini-Xmodem, and many others, without loading from tape each time. Furthermore, it is compatible with Mr. Rigger's (The ZED Group) bit-mapped HI-RES. (More on this later-ed.)

## How to Do it

You do not have to be a technical wizard to implement this project. No trace cuts are required on the computer board. We will build a small sub-assembly which plugs into the original 24-pin RAM

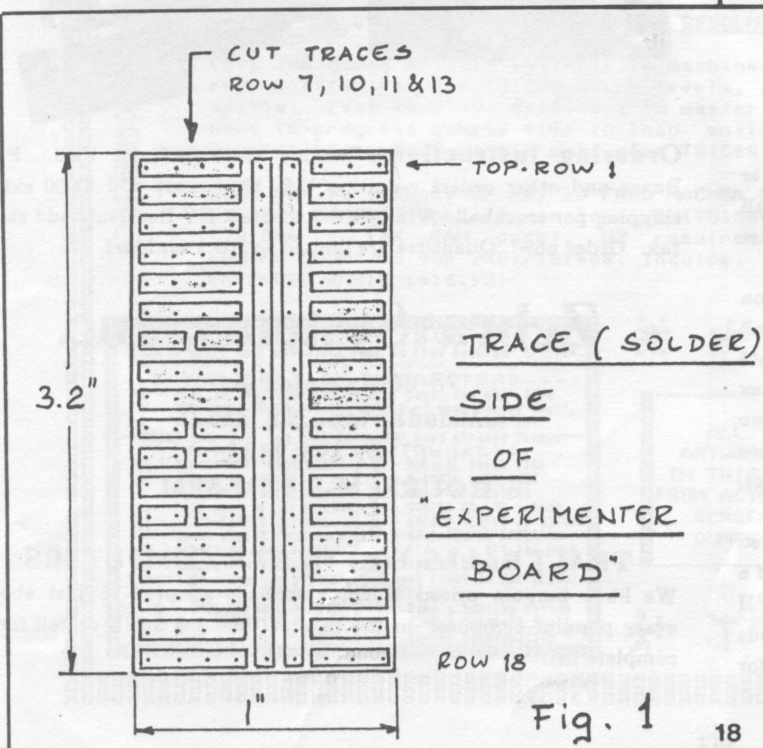
socket. Seven wire leads are then soldered to the computer board, and a battery holder is mounted using double-sided adhesive foam. Two additional wires connect to the battery.

If your particular computer has the 2K RAM soldered in, your best option is to scout around for another board with a socketed 2K RAM chip. Alternately, clip out the 2K chip with small, sharp wire cutters. Then remove the "legs" with needle-nose pliers and a soldering iron. Finally, use a suction-type solder remover to clean out the holes, and install a 24-pin socket.

If you have the original ZX81 with two 1K chips soldered in, you can simply leave them in. Solder a 24-pin socket into the space marked on the board for it. You will have to cut the center support "strut" to clear the 1K chip that lives in the center of the 24-pin socket pattern.

Obtain an "Experimenter Board" from Radio Shack, and cut out a 1" X 3.2" piece as shown in Figure 1. Note that Fig. 1 is the view from the circuit (trace) side. Insert a low-profile 28-pin wire-wrap socket as shown, letting the leads protrude 3/16" on the trace side. Solder the lower 24 pins (only) to the board. These will be the "legs" that plug into the 24-pin socket on the computer, so be careful to keep the pins free of excess solder.

Now use SMALL and SHARP wire cutters to cut the 24 soldered pins on the "component" side, and set the socket aside. Then cut four traces between the socket and the male legs, as shown in Fig. 1, at pin numbers 20, 22, 23 and 26 of the 28-pin socket. (Note that the numbers apply to the 28-pin socket that will later be re-installed.)



### SHOPPING LIST

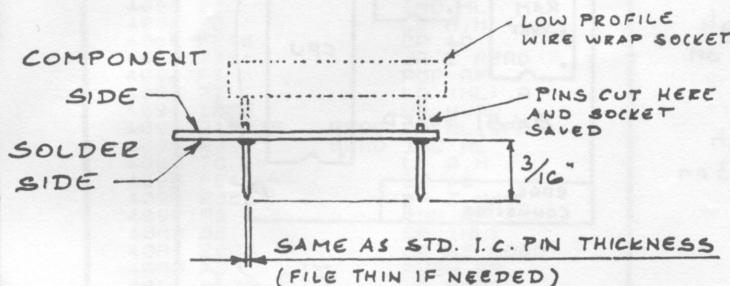
- "Experimenter Board" Radio Shack #26-150 or equivalent
- U1-HM 6264 LP-15 (Hitachi or Equal)
- 8Kx8 Static RAM
- R1-1K $\Omega$  1/8W carbon film resistor
- R2-1M $\Omega$  1/8W carbon film resistor
- D1-Schottky or Germanium type
- Signal Diode
- D2-D5 1N4148 Diodes
- Q1-2N3904 NPN Transistor
- B1-3 Volt Battery System: 2 "AAA" Alkaline cells complete with holder and double sided foam tape

Fig. 1



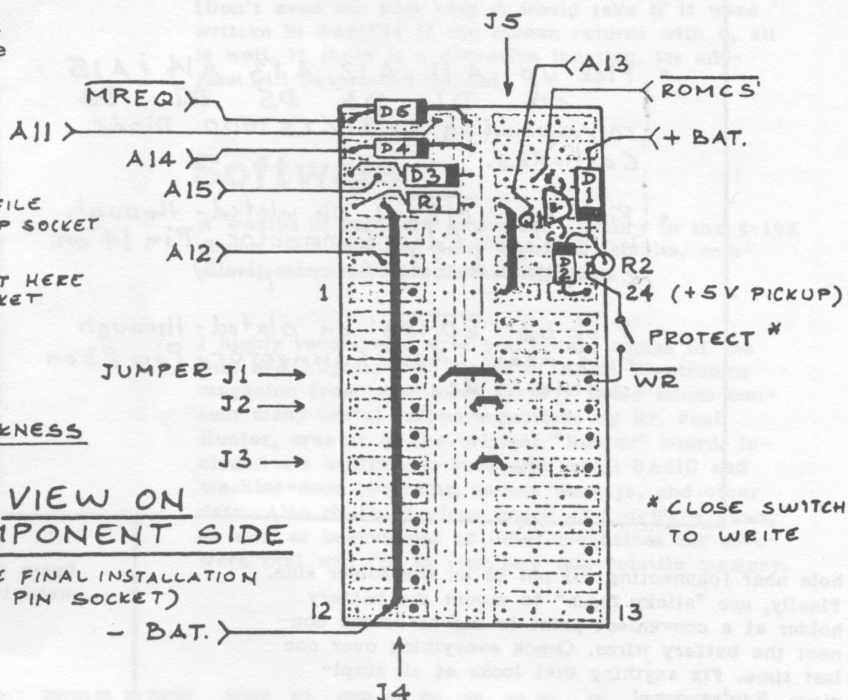
#### ASSEMBLY SEQUENCE:

1. Jumpers (5)
2. Diodes (5) Do Not solder anode lead of D2 yet
3. The 26 pins off the 28 pin socket. Solder anode lead of D2
4. Resistors (2)
5. Transistor
6. The flying leads
7. See FIG. 2b: the 28 pin socket



**VIEW ON COMPONENT SIDE**  
(BEFORE FINAL INSTALLATION OF 28 PIN SOCKET)

Fig. 2 a



Next we'll install the components and jumpers as indicated in Figure 2a.

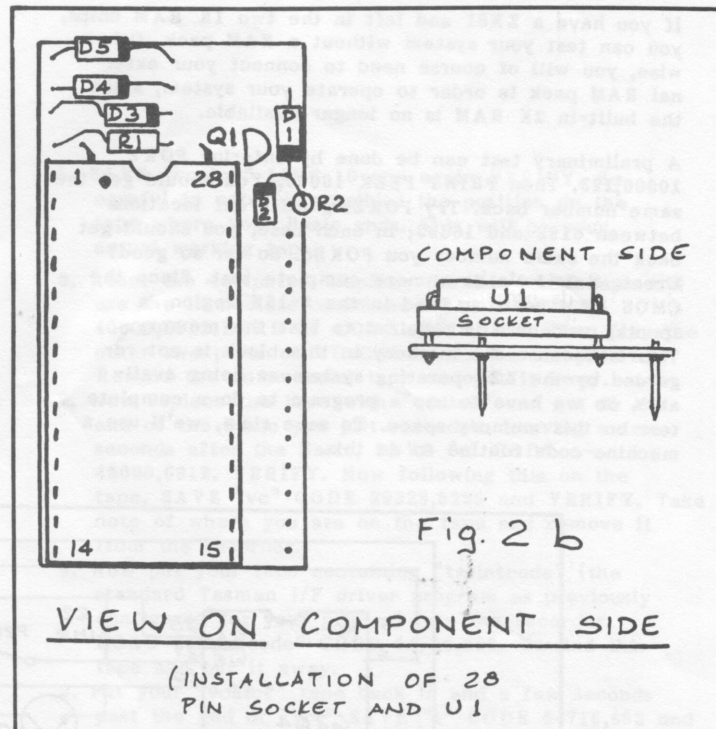
Install jumpers, components and "flying leads," referring to Fig. 2a, and the schematic of Fig. 3. To make final assembly easier, you might want to color-code the flying leads. Individual strands of colored ribbon cable are one possible source of small-gauge colored wire. Note that diode D1 should preferably be a Schottky or Germanium type; silicon (1N4148) works on my unit, but might have too much forward drop to give reliable battery back-up with some chips.

Now insert the just-freed 28-pin socket into the board; note that the socket is offset from the "legs" by two holes. (See Fig. 2b.) Solder all 28 pins, and cut off the excess lengths.

If you wish to install a write-protect switch, connect flying leads to the points shown. When this line is open, the board will be write-protected. If you don't want this feature, replace this line with a jumper. Check your work, and double-check it. Now check it again. Be alert for shorts, "cold" (dull-looking) solder joints, and backwards diodes.

## Final Wiring

(NOTE: Before we get on with the installation, a warning is in order. If you have moved your computer to a larger case, with plenty of headroom over the 2K socket, then you have nothing to worry about. However, if your machine is still in the stock case, then the installation of this addition would kink and thereby break the traces on the larger keyboard "tail." More headroom is required over the new installation to allow for the already minimal bend radius in the ribbon cable. Install spacers, 1/2" long, between the component side of the board and the case top. Ideally, a grounded metal skirt should be installed around the resulting perimeter gap, to contain RFI and keep out dust. You might find another solution, such as shortening the cable slightly and/or taping it to curve the other way.)



Plug your newly-built module into the 24-pin 2K RAM socket. Connect the seven flying leads to the ZX81 board. The best place to pick up the address lines A11 through A15 is at the cathode (banded) ends of the keyboard diodes. See Figure 2c. The diodes are number D1 through D8, starting at the end closest to the ROM chip. "A11" goes to D1, "A12" to D3, "A13" to D5, "A14" to D7, and "A15" to D8. Be careful about shorts, as there isn't much clearance between the diode leads. Pick up MREQ NOT at the plated-through hole near (and connecting to) edge connector pin 14, component side. (Remember, the keyslot is "pin 3.") Pick up ROMCS NOT at the plated-through

- Pick up A11, A12, A13, A14 & A15 on D1, D3, D5, D7, D8 corresponding ZX81/TS1000 Diode Cathodes.
- Pick up MREQ on plated-through hole near Edge Connector - Pin 14 on component side (Key is pin 3)
- Pick up ROMCS on plated-through hole near Edge Connector - Pin 23 on solder side.

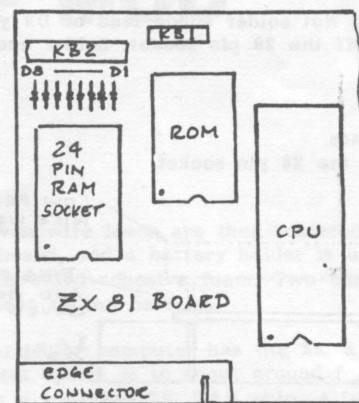


Fig. 2 c

hole near (connecting to) pin 23 on the solder side. Finally, use "sticky foam" to mount the battery holder at a convenient point in the case, and connect the battery wires. Check everything over one last time. Fix anything that looks at all suspicious. You're done!

## Testing

If you have a ZX81 and left in the two 1K RAM chips, you can test your system without a RAM pack. Otherwise, you will of course need to connect your external RAM pack in order to operate your system, since the built-in 2K RAM is no longer available.

A preliminary test can be done by entering POKE 10000,123. Then PRINT PEEK 10000. You should get the same number back. Try POKEing different locations between 8192 and 16383; in each case, you should get back the same number you POKEd. So far so good? Great. Now let's do a more complete test. Since the CMOS RAM board is used in the 8-16K region, a special procedure is required to test the memory. This is because the memory in this block is not regarded by the ZX operating system as being available. So we have to use a program to do a complete test on this memory space. To save time, we'll use a machine-code routine to do this.

Enter the following short BASIC program, which will make it easy to enter the machine-code test routine:

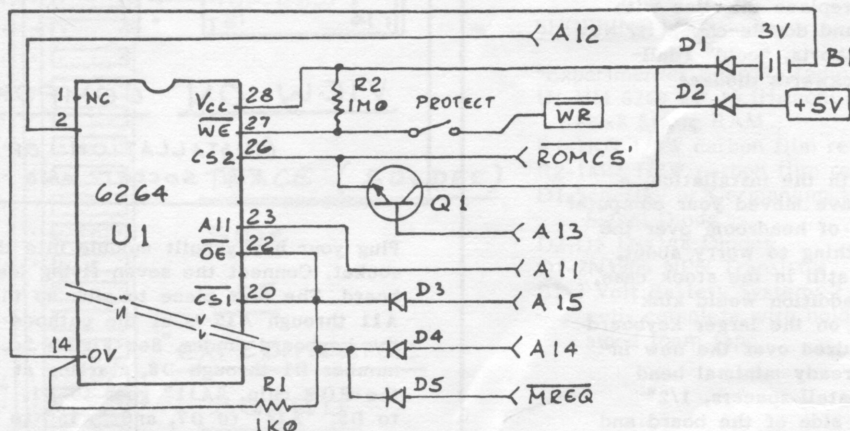
```
MACHINE-CODE LOADER FOR TEST
1 REM 12345678901234567890123
45678901234567890123
10 FOR A=16514 TO 16556
20 SCROLL
30 INPUT N
40 POKE A,N
50 PRINT PEEK A
60 NEXT A
```

Enter RUN, and input the values from the following table, going from left to right, top to bottom. If you make a mistake, enter STOP, then LET A=A-1, then GOTO 20, and re-input the correct number.

TEST ROUTINE: DECIMAL VALUES

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 175 | 245 | 241 | 61  | 40  | 33  | 33  | 255 |
| 31  | 245 | 35  | 124 | 254 | 64  | 40  | 4   |
| 241 | 119 | 24  | 245 | 33  | 255 | 31  | 35  |
| 124 | 254 | 64  | 40  | 229 | 241 | 190 | 32  |
| 3   | 245 | 24  | 243 | 68  | 77  | 201 | 1   |
| 0   | 0   | 201 |     |     |     |     |     |

This installs a machine-language routine (by Fred Nachbaur), which tests every memory location from 8192 to 16383, with every possible value from 0 to 255. (A total of 2,097,152 writes and reads!) The disassembly is shown.



SCHEMATIC

Fig. 3



Enter FAST mode, then enter PRINT USR 16514. If all is well, the routine will take about 90 seconds to

```

4082 AF      TEST XOR A
4083 F5      PUSH AF
4084 F1      NXUL POP AF
4085 3D      DEC A
4086 2321    JR Z OKAY
4088 21FF1F  WRIT LD HL,1FFF
408B F5      NWAD PUSH AF
408C 23      INC HL
408D 7C      LD A,H
408E FE40    CP 40
4090 2804    JR Z READ
4092 F1      POP AF
4093 77      LD (HL),A
4094 18F5    JR NWAD
4096 21FF1F  READ LD HL,1FFF
4099 23      NRAD INC HL
409A 7C      LD A,H
409B FE40    CP 40
409D 28E5    JR Z NXUL
409F F1      POP AF
40A0 BE      CP (HL)
40A1 2003    JR NZ NOGD
40A3 F5      PUSH AF
40A4 18F3    JR NRAD
40A6 44      NOGD LD B,H
40A7 4D      LD C,L
40A8 C9      RET
40A9 010000  OKAY LD BC,0000
40AC C9      RET

```

run. It would take about ten minutes in SLOW mode. (Don't even ask how long it would take if it were written in BASIC!) If the screen returns with 0, all is well. If there is a defective location, its address will be printed instead.

## Software

A wealth of software exists for memory in the 8-16K region. Included are many types of toolkits, compilers, assemblers, and other utilities.

I highly recommend that you obtain copies of the July and August, 1983 issues of Radio-Electronics magazine from your local library. These issues contain many useful software routines by Dr. Paul Hunter, creator of the original "Hunter" board. Included are utilities to save and recall BASIC and machine-code programs, screen displays, and other data. Also check previous issues of SyncWare News, as well as back-issues of other magazines for software that will run in your new non-volatile memory.

# VU-CALC & THE TASMAN I/F

Pat Morrissey (415) 952-5068  
2000 Crystal Springs Road  
Bldg 21, Apt 22  
San Bruno, CA 94066

Here is a short program which will allow you to use your "80 Column" Printer to print a real spread sheet for presenting data (including tables you've already SAVED) from the Psion Software program VU-CALC. Required hardware and software are:

1. T/S 2068 Computer
2. Tasman Centronics Printer I/F
3. 80 Column Printer
4. Tape Recorder
5. VU-CALC
6. Tasman printer software - "tasintcode" - properly customized for your printer.

Below is a set of Basic program lines which are to be MERGED with the Basic portion of VU-CALC. To do this, proceed as follows:

1. Reset the computer and key in the lines listed herein.
2. SAVE and VERIFY them on a bank cassette. This is only a transfer tape and will not be the final "product," so use any name you like.
3. Reset the computer and MERGE the Basic part of VU-CALC. (That's the first part on the tape.) You must use MERGE to prevent the program from Auto starting. Stop the tape without rewinding it immediately after the border shows that the program is finished loading. Without rewinding the tape, remove it from the recorder. This will set the tape at the right place for loading in the other portions of VU-CALC later.
4. LIST the VU-CALC Basic. Use the Immediate Command MERGE". Merge in the previously Saved lines from 1 & 2 above.

5. SAVE "vcalcp" LINE 10 and again VERIFY. Be careful to note (if possible) the position on the tape where your Basic ends. This will be your actual working tape.
6. Reset the computer. Replace the VU-CALC tape and use the immediate command LOAD "" CODE 40000,6912:LOAD "" CODE 29328,5225. Then play the other two parts of VU-CALC into the computer. Rewind the commercial tape and put it away.
7. Now replace the tape with "vcalcp" and get to the end of the Basic you just recorded. Leaving a few seconds after the Basic, SAVE "s" CODE 40000,6912. VERIFY. Now following this on the tape, SAVE "vc" CODE 29328,5225 and VERIFY. Take note of where you are on the tape and remove it from the recorder.
8. Now put your tape containing "tasintcode" (the standard Tasman I/F driver program as previously configured for your printer) into the recorder. LOAD "tasintcode" CODE 64716,652. Rewind this tape and put it away.
9. Put your "vcalcp" tape back in and a few seconds past the end of "vc," SAVE "t" CODE 64716,652 and VERIFY.

That's it! Now, reset the computer and use LOAD"" to test the program for proper operation.

### NOTES:

- a) This program was written for use on the Star SG-10 Printer. Lines in the program which may require modification for your printer are: 51, 53, 60, 63, 69, 145 and 180. With only one change (same in LINES 51 & 63) the program runs on the Blue Chip M120/10.

```

10 CLEAR VAL "29327": BORDER L
N PI: PAPER LN PI: INK LN PI: CL
S : LOAD ""SCREEN$: LOAD ""CODE
: LOAD ""CODE: GO TO VAL "3200"

50 RANDOMIZE USR VAL "64719"
51 LPRINT CHR$ VAL "27":CHR$ U
AL "27":CHR$ VAL "77":CHR$ VAL "
27":CHR$ VAL "0"
52 INPUT "Title: ";a$
53 LPRINT CHR$ VAL "27":CHR$ U
AL "14":CHR$ VAL "a$
60 LPRINT CHR$ VAL "27":CHR$ U
AL "15"
62 INPUT "Left Margin (Condens
ed) ";s
63 LPRINT CHR$ VAL "27":CHR$ U
AL "27":CHR$ VAL "77":CHR$ VAL "
27":CHR$ s
68 INPUT "Start Column ";n: IF
n<1 THEN LET n=1
69 INPUT "Finish Column ";m: I
F m>(n+18-(INT ((s+6)/7))) THEN
LET m=n+18-(INT ((s+6)/7))
70 INPUT "Start Line ";q: IF q
<1 THEN LET q=1
71 INPUT "Finish Line ";r: IF
r>50 THEN LET r=50
72 IF m>50 THEN LET m=50
73 FOR a=1 TO (m-n+1)
75 LPRINT TAB (1+(a*7)-LEN STR
$(a+n-1));a+n-1;
80 NEXT a
90 LPRINT
120 FOR i=((q-1)*350) TO ((r-1)
*350) STEP 350
125 LPRINT CHR$ VAL "32";
130 FOR k=((n-1)*7) TO 350
140 LPRINT CHR$ PEEK (34573+k+i
)
145 IF k=((m*7)-1) THEN LET k=3
50: LPRINT CHR$ VAL "27":CHR$ VA
L "13"
150 NEXT k
160 NEXT i
173 FOR a=1 TO (m-n+1)
175 LPRINT TAB (1+(a*7)-LEN STR
$(a+n-1));a+n-1;
178 NEXT a
179 LPRINT
180 LPRINT CHR$ VAL "27":CHR$ U
AL "18"
190 RETURN
2000 GO TO USR a3
3000 GO SUB 1200: PRINT AT 9,2:"
ENTER 1 EXIT PROGRAM""TAB 9;
"2 : CLEAR WORKSHEET""TAB 9;"3
: RETURN TO VU-CALC""TAB 9;"4 :
PRINTOUT": INPUT "OPTION? ";a
3010 IF a>0 AND a<5 THEN GO TO (
3000+a*100)
3200 CLEAR 29327: DIM b$(100): D
IM c$(20): GO SUB VAL "1200"
3400 CLS : GO SUB 50: GO TO 3000
4000 GO SUB VAL "1100": SAVE a$c
ODE zz:(PEEK bfre+256*PEEK (bfre
+1)-zz): CLS : PRINT AT 5,1:"Rw
ind and Play Tape to Verify""
GO TO 3000 ON ERROR AND RESAVE":
VERIFY ""CODE: CLS : GO TO USR
a2

```

(c) 1985 C.P. Morrissey. All rights reserved. Do not copy or reprint in whole or part without express written permission of the author. Commercial use strictly prohibited.

- b) With only minor modifications, this program will also drive "136 Column" printers (e.g. Star SG-15, Epson FX-100). This should give you the capability to print up to 32 columns of VU-CALC data!
- c) Although there is room to add a few more Basic lines to the program, I have not put in the usual REMark lines to properly document the revisions internally. I present here the notes which have been omitted from the program to conserve memory:

50 Initialize "t" ("tasintcode") and printer.  
51 Initialize left margin to 0. The number "77" herein is one code that doesn't work with the Blue Chip. Change it to "108".

- 53 Commands "One line expanded print", prints a\$ (title).  
60 Commands "Condensed print."  
63 Commands "Set left margin = s". See Note for LINE 51.  
69 Checks for more columns than will fit sheet. If you use a print font other than "condensed" (17 cpi) you will likely want to change this the number 18 to a smaller value. See Note e) below. It is also where you would put in a larger value for a "136 Column" printer.  
73-80 and 173-180 print the column numbers at the top and bottom, respectively, of the hard copy. Delete or change these to suit your needs. I like the top and bottom system, because it gives me a place to put my ruler when I draw column separating lines.  
120-160 The data "fetch"/LPRINT routine. Good luck with modifications. One idea that might be worth the effort is a change to leave spaces between columns. I prefer the extra columns of printout afforded by the existing omission of the spaces. This also avoids splitting text that extends over more than one column.  
180 Commands "Pica print" (Stops Condensed print).  
2000 See Note d) below.  
3200 See Note f) below.

- d) Line 2000 disables the "#p" command accessed from the spreadsheet data display. Adding Line 2000 simply causes the program to ignore "#p." If you like, don't include line 2000 below (i.e., leave the original line in VU-CALC alone). That will allow you to alternate between printers at your command. Just be sure that if you use "#p", your T/S 2040 is connected. If this command is invoked, as in the original program, without the T/S 2040 Printer connected, the program will appear to hang. You can get back in action without losing data by using BREAK, GO TO 3000 and then "3: Return to VU-CALC". Since making these modifications to the program I've found the T/S 2040 superfluous.
- e) The quality of print from the SG-10 is sufficient to allow use of condensed print for hard copy. If your printer doesn't have condensed print or isn't clear enough in that mode, you will get fewer columns per report. The sample herein shows 19 columns, the max I can squeeze on a standard page. Change Line 69 from 18 to one less than your required maximum number of columns.
- f) Line 3200 has been listed before as a cure for the "2: Clear Worksheet" bug in the commercial program. See TIMELINEZ Vol. 2, No. 5 - May 84 - Page 34. If it's already part of your program, you don't need to re-enter it here.
- g) I've also added a VERIFY routine to check data SAVED from the Program.

For those of you with modems that can download programs, this program is available at the number listed above. It's your nickel.

During the development of the program modifications, a number of bugs jumped up from unexpected corners. Those bugs are gone. But, if you find others PLEASE let me know. Thanks.

\*\*\*\*\*  
**SYNCLARE**  
SYNCLONDS



# BBDOS:

## TS1000 Review

BBDOS: A Disk Operating System  
for the AERCO FD-ZX Interface

Distributed by:  
Bill Bell

596 Cherrington Road  
Westerville, OH 43081  
(614) 882-3883  
PRICE: \$29.95

ZX81/TS1000 owners who have been using the Aerco FD-ZX Disk Drive Interface owe Jerry Chamkis and his Texas staff a debt of gratitude for providing them with a reliable, high speed, mass storage system. But even as nice as the FD-ZX unit is, its disc operating system (DOS) leaves much to be desired. The machine code part of the DOS stored in an on-board EPROM performs its job well enough, but the BASIC program that drives it is, to be polite, caveman-ish in its approach.

Enter Bill Bell of the Columbus, Ohio area.

Mr. Bell picked up his first AERCO FD-ZX in a computer "junk" store somewhere on the Eastern Seaboard while on a trip there a few years back. He liked the hardware and the firmware, but was disappointed with the BASIC DOS driver. Which led to the question, "What if...".

Mr. Bell is now offering for sale his disk operating system for the AERCO FD-ZX. The system is BBDOS (appropriately for Bill Bell Disk Operating System). This DOS will take your ZX81/TS1000 and FD-ZX Interface and propel them from the pre-Neanderthal era into the 21st century.

BBDOS is fully automatic and BASIC transparent. The only requirement for its use is that there must be memory available and enabled in the 2000-2FFF (hex, 8K-12K decimal) region of the computer memory map. Once the DOS is installed, it is completely immune

to BASIC "NEW" and even system reset (pin 21B of the rear edge connector brought low). To install and start the DOS requires only one RAND USR call. After that, all file handling is done by file name or a function code from the DOS directory. This arrangement completely frees the user from the tedious and mundane chore of disk management.

Other features include automatic handling of 16K/64K page intermixing. Easy conversion of programs and data previously stored with SADOS (the AERCO DOS package). Single file and full disk transfers or moves (copy). BBDOS even allows you to copy files from disk to disk with only a single disk drive connected to the FD-ZX. BBDOS also permits the auto-booting of a selected program from disk upon DOS startup. Provisions are made in the DOS to allow 45 functions to be easily initiated by your own BASIC programs (with just a little effort, your MC programs can even do it!-ed.). By special arrangement with Fred Nachbaur, Memotech, and Ray Kingsley, Memotext, Z-Tools, and Hot Z II can be provided on disk already modified to function with BBDOS.

BBDOS supports 1 or 2 drives, single or double sided. It also supports all printer interfaces currently on the market for the ZX81/TS1000.

While the documentation is not exhaustive, it is more than adequate to get you up and running and completely acquainted with all the features of BBDOS. If any problems or questions do arise, this author has first hand knowledge that Mr. Bell is more than willing to take the time to talk or correspond with you until you both are assured your questions have been answered fully.

I would highly recommend the purchase of BBDOS for use with your AERCO FD-ZX interface. It takes a good product and makes it even better.

Reviewed by Jeff Moore

## Ireg 1000

Wes Brzozowski

The TS1000 owners manual states (p. 123) that upon return from machine code, the I register must have the value 1Eh. However, it can be a lot of fun breaking the rules. The I register points to the upper byte of the character generator table in the ROM. If we change it, the character set changes to a random hash, which can be a useful effect.

This program works by setting a basic variable, IREG, equal to what we want in the I register. RAND USR 16514 will change the value in the I register. When this register is changed, the screen turns to chaos, which is great for getting peoples attention and simulating explosions.

Only the 1 REM line is necessary for the program to operate, so after you enter and test the program, you may delete all of the other lines. The REM line must contain at least 38 spaces. Use your favorite POKer program to enter the data. To make this routine relocatable, you would have to change the first instruction (LD HL, 40A3) to point to the name

of the variable that you would like to look up the value of (IREG in this case). If you would like to change the name of the variable, then you can change the data at address 40A3 to any variable name that you desire. Just end the name with a + sign.

The Basic program listing has several REM lines containing the source listing of the machine code. It is not necessary to enter these lines into the program.

Enter the data left to right and top to bottom.

### DECIMAL POKE TABLE

|     |     |    |     |     |     |     |     |
|-----|-----|----|-----|-----|-----|-----|-----|
| 33  | 163 | 64 | 34  | 22  | 64  | 14  | 0   |
| 205 | 28  | 17 | 48  | 2   | 207 | 1   | 35  |
| 237 | 91  | 28 | 64  | 205 | 246 | 25  | 235 |
| 34  | 28  | 64 | 205 | 205 | 21  | 237 | 71  |
| 201 | 46  | 55 | 42  | 44  | 21  |     |     |

# CLASSIFIED

Do you have something to sell?  
Let everyone know. Just \$2.75 a  
line will put it here. 32 column  
(screen) format, include spaces  
in your text, and send CK. or MO  
with your listing.

Send to SWN Classified, 602 S.  
Mill St., Louisville, OH 44641.

**NEEDED:** One (1) video monitor as  
donation to church (computer and  
printer already donated).

Contact Floyd Rathbone, 1st  
Baptist Church of Berwick, 3919  
Mount St., Berwick, LA 70342.  
Phone (504) 384-5241.

**JUNKWARE** from Thomas B Woods!  
Over the past 5 years we have  
collected hundreds of tapes  
which were returned for one  
reason or another. Most load.  
Most have genuine TBW titles. At  
worst, you get perfectly usable  
cassettes at a great price. And  
maybe, you'll get a real bargain!  
No Guarantees. You buy AS IS.  
But look at the price!!  
6 tapes just \$3 WOW!! The box is  
overflowing. we've got to get  
'em out of here. Add \$1 postage.  
TBW, Box 64, Jefferson, NH  
03583.

## IREG Basic Listing

```

1 REM 5-RND6-RND: LN 0)K*INT
*7 GOSUB ?ORNDLN PLOT: FOR 60RN
DLN LN+ GOSUB ?TAN IREG+
110 REM LD HL, /1000
120 REM LD (H4016), HL
130 REM LD C, 0
140 REM CALL H111C
150 REM JR NC, /180
160 REM RST 08
170 REM DATA H01
180 REM INC HL
220 REM LD DE, (H401C)
230 REM CALL H19F6
240 REM EX DE, HL
250 REM LD (H401C), HL
260 REM CALL H15CD
270 REM LD I, A
280 REM RET
1000 REM DATA "IREG+"
2000 REM END
4000 LIST 110
5000 FOR J=0 TO 30 STEP 2
5010 LET IREG=J
5020 RAND USA 16514
5025 NEXT J

```

## IREG Disassembly

```

4032 21A340 LD HL, 40A3
4035 221640 LD (CHADD), HL
4038 0E00 LD C, 00
403A CD1C11 CALL 111C
403D 3002 JR NC 4091
403F CF RST 08H
4040 01 ERROR 2
4041 23 INC HL
4042 ED5B1C40 LD DE, (STKND)
4045 CDF619 CALL 19F6
4049 EB EX DE, HL
404A 221C40 LD (STKND), HL
404D CDCD15 CALL 15CD
4040 ED47 LD I, A
40A2 C9 RET
40A3 2E37 LD L, 37
40A5 2A2C15 LD HL, (152C)
40A8 76 HLT
40A9 00 NOP
40AA 6E LD L, (HL)
40AB 0D DEC C
40AC 00 NOP

```

# SyncWare News

## CONTENTS

|                                |    |
|--------------------------------|----|
| For Your Support....           | 2  |
| Forum....                      | 3  |
| TS1000 Error Reports....       | 7  |
| Hoffman                        |    |
| Redefine the Character Set.... | 9  |
| (TS2068) Majors                |    |
| Basil's Compendium....         | 13 |
| Wentworth                      |    |
| One Chip Mod/A Built-In        |    |
| Non-Volatile Memory....        | 18 |
| Breunung (TS1000)              |    |
| Vu-Calcul & The Tasman I/F.... | 21 |
| (TS2068) Morrissey             |    |
| BBDOS: TS1000 Review....       | 23 |
| Ireg: TS1000....               | 23 |
| Brzowski                       |    |

FIRST-CLASS MAIL  
U.S. POSTAGE  
PAID  
Jefferson, NH  
Permit No. 1

Z.A. Lewis  
PO Box 559  
Camino, CA

95709

4:4

Editor:  
Jeffrey Moore  
602 S. Mill Street  
Louisville, OH 44641

Technical Director:  
Fred Nachbaur  
C-12, Mtn. Stn. Broup Box  
Nelson, BC V1L 5P1 Canada

Advertising, Subscriptions:  
Thomas B. Woods  
P.O. Box 64  
Jefferson, NH 03583

Submissions, announcements, letters to the editor,  
and all other material of editorial interest should  
be sent to the Ohio address.

Copyright 1986 SyncWare News

# Subscribe! \$16.95

For Canada and Mexico please add \$3.00 postage.  
All other foreign add \$9.00 Airmail (U.S. funds only)

1 year (6 issues)